Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Diplomarbeit
cand.-Ing.
W. Andreas Klimke

T-Profil 50x50x15x15
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Diplomarbeit
cand.-Ing.
W. Andreas Klimke

Betreuer: Prof. Dr.-Ing. Stefan M. Holzer
Fachgebiet Informationsverarbeitung im Konstruktiven Ingenieurbau
Institut für Baustatik, Universität Stuttgart

© 1998 W. Andreas Klimke
Fünf-Bäume-Weg 191
89081 Ulm
Vorwort


Aufgrund des großen thematischen Umfangs sind die einführenden Kapitel etwas knapper gehalten. Ich habe mich hier auf die für das Verständnis dieser Diplomarbeit notwendigen Grundlagen beschränkt. Weiterführende Literaturhinweise zur St.-Venantschen Torsion, sowie zur Randintegralmethode sind im Anhang zu finden.


An dieser Stelle sei auch Herrn Martin Bernreuther herzlich gedankt, der als kompetenter Ansprechpartner bei beliebigen computerfachlichen Anliegen zur Verfügung stand.
Inhaltsverzeichnis

1 Einführung ........................................................................................................................................ 1
  1.1 Gegenstand der Arbeit............................................................................................................. 1
  1.2 Eine kurze Erklärung des Begriffs „Randintegralmethode“ .................................................. 1
2 Die Membrananalogie .................................................................................................................. 3
  2.1 Allgemeine Membran .......................................................................................................... 3
  2.2 Einführung wichtiger Membran-Gleichungen .................................................................. 4
  2.3 Zusammenhang mit der Torsion ....................................................................................... 5
  2.4 Reduktion des Membranproblems .................................................................................... 6
3 Implementierung der Randelementmethode ........................................................................... 7
  3.1 Übersicht über die Vorgehensweise .................................................................................. 7
  3.2 Datenstruktur zur Erfassung der Membrangeometrie ...................................................... 7
  3.3 Diskretisierung des Randes .............................................................................................. 9
  3.4 Schritte zur Lösung des Randwertproblems mit Hilfe der Kollokation ......................... 11
    3.4.1 Aufstellen des Gleichungssystems ........................................................................... 12
    3.4.2 Lösen der Integrale ................................................................................................. 13
    3.4.3 Lösen des Gleichungssystems ............................................................................... 14
  3.5 Programmierung der Kollokation in C++ ...................................................................... 15
  3.6 Möglichkeiten zum Testen der Ergebnisse und Auswertung ........................................... 17
  3.7 Aufbringen der Gebietslast: gleichmäßiger Druck ......................................................... 21
4 Berechnung des Torsionswiderstandsmoments .................................................................. 23
5 Fehlergrößen ............................................................................................................................. 29
  5.1 Fehler bezüglich des Kräftegleichgewichts .................................................................. 30
  5.2 Fehler bezüglich der Verschiebung: $u^2$ - Residuum .................................................... 33
6 Adaptive Randelement-Teilung .............................................................................................. 37
  6.1 Residuum der Verschiebung als Kriterium für Adaption ................................................. 37
  6.2 Programmierung der Adaption ....................................................................................... 38
  6.3 Beurteilung des adaptiven Vorgehens ............................................................................ 39
7 Das Galerkin-Verfahren ......................................................................................................... 44
  7.1 Realisierung des Galerkin-Verfahrens ............................................................................. 45
  7.2 Bewertung ....................................................................................................................... 48
8 Querschnitte mit Löchern ........................................................................................................ 55
8.1 Mehrere Ränder ............................................................................................................ 56
  8.1.1 Liste zur Verwaltung der Ränder ........................................................................... 57
  8.1.2 Kollokation mit mehreren Ringlisten ................................................................. 57
  8.1.3 Überprüfung der Berechnungen .......................................................................... 61
8.2 Vom Membranvolumen zum Torsionswiderstandsmoment ........................................ 63
  8.2.1 Methode nach Griffith und Taylor ....................................................................... 64
  8.2.2 Hinweise zur Programmierung .......................................................................... 67
  8.2.3 Ergebnisse für die Beispielquerschnitte ............................................................... 70
9 Performance-Vergleich: Finite Elemente – Randelemente ........................................... 73
  9.1 Vergleich bezüglich Rechenzeit .............................................................................. 73
  9.2 Vergleich weiterer typischer Eigenschaften ............................................................. 76
10 Verbesserung der Ergebnisse durch Extrapolation .................................................... 78
11 Einbettung der BE-Methode in eine graphisch-interaktive Umgebung ...................... 82
  11.1 Die Basisbibliothek CIALib .................................................................................... 82
  11.2 Aufbau des graphisch-interaktiven Programms ...................................................... 84
  11.3 Programmbeschreibung: BEM Torque Calculation .............................................. 85
    11.3.1 Erstellen neuer Modelle .................................................................................... 85
    11.3.2 Anpassen der Anzeige .................................................................................... 87
    11.3.3 Laden und Speichern von Modellen ............................................................... 87
    11.3.4 Ausführen von Berechnungen ....................................................................... 87
    11.3.5 Sonstige Funktionen ..................................................................................... 89
12 Zusammenfassung ........................................................................................................ 90
13 Anhang ........................................................................................................................ 91
  A Inhalt der Begleit-CD ............................................................................................... 91
  B Literaturhinweise ....................................................................................................... 92
  C Abbildungsverzeichnis ............................................................................................. 92
1 Einführung

1.1 Gegenstand der Arbeit


Ich möchte in dieser Arbeit neben der mathematischen Lösung des Problems, die in weiten Teilen bereits sehr gut in Fachliteratur beschrieben ist (z.B. [1]), auch auf die Aspekte der Implementierung benötigter Datenstrukturen und Algorithmen eingehen.


1.2 Eine kurze Erklärung des Begriffs „Randintegralmethode“

Da die „Randintegralmethode“ bei Ingenieuren im Vergleich zur Methode der finiten Elemente weniger bekannt ist möchte ich für den interessierten Leser, der mit diesem Begriff nichts anfangen kann, eine kurze Definition versuchen.


Ein einfaches Beispiel im eindimensionalen Fall wäre z.B. ein Geradenstück. Durch die Randwerte \( a \) und \( b \) ist es eindeutig bestimmt.
Die Einflußfunktion ist dann eine Geradengleichung \( f(x) = ax + b \), die es erlaubt, alle zunächst unbekannten Geradenpunkte „im Inneren“ zu bestimmen (siehe Abbildung 1).


Im zweidimensionalen Fall ist die Sachlage ähnlich – der Rand besteht hier aus einer das Gebiet begrenzenden Kurve. (Im Dreidimensionalen stellt der Rand dann eine Oberfläche dar.) Auch hier existieren Einflußgleichungen, die die Bestimmung der Kraft- und Weggrößen beliebiger Innenpunkte ermöglichen.

Zusammenfassend läßt sich für alle Fälle sagen, daß das Problem gelöst ist, sobald alle Randgrößen bekannt sind. Dann müssen die Randdaten nur noch in die Einflußfunktionen eingesetzt werden, um Knotenverschiebungen/Spannungen im Innern des Bauteils zu erhalten. Dies führte in der Fachliteratur zur Bezeichnung Randwertproblem.

In der Regel kennt man genau die Hälfte der Randbedingungen (siehe [1], S.11f); die andere Hälfte muß erst berechnet werden. Dies ist prinzipiell stets möglich, da die unbekannten Randwerte zu den bekannten konjugiert sind. Hierfür sind verschiedene Ansätze möglich: Kollokation und Galerkin werden im Rahmen dieser Diplomarbeit behandelt.
2 Die Membrananalogie


2.1 Allgemeine Membran

- Stichwortartig zusammengefaßt wird eine Membran wie folgt definiert:
  - Die Membran stellt eine ebene Fläche in Form eines elastischen Gewebes dar. Sie ist zweidimensional, d.h. sie hat keine nominelle Dicke.
  - Am Rand ist sie teils fest, teils frei gelagert und dabei mit einer konstanten Normalkraft vorgespannt.
  - Die Verformungen sind senkrecht zur Membranfläche, genauso wie die Belastungen.
  - Materialeigenschaften spielen bei der Membrantheorie keine Rolle (es wird keine Dicke angenommen).

Ein Beispiel für eine Membran unter diversen Belastungen zeigt Abbildung 2.

![Abbildung 2: Beispiel für eine beliebige Membran](image-url)
2.2 Einführung wichtiger Membran-Gleichungen

Zur Notation: Bei den fettgedruckten Symbolen handelt es sich um Vektoren.

Die Membran sei mit einer Kraft \( N \) vorgespannt und mit dem gleichmäßigen Druck \( p \) belastet. Dann gilt für jeden Punkt \( x = (x_1, x_2) \):

\[
\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = -N\Delta u = p.
\]

Außerdem gilt für die Spannung \( t \) im Punkt \( x \), wobei \( n \) die Schnittnormale darstellt:

\[
t = N \frac{\partial u}{\partial n} = N \left( \frac{\partial u}{\partial x_1} n_1 + \frac{\partial u}{\partial x_2} n_2 \right).
\]

Die Einflußgleichung der Membran für die Verschiebung \( u(x) \) lautet:

\[
c(x)u(x) = \int_{\Gamma} \left[ g(o(y, x)N \frac{\partial u}{\partial n}(y)) - N \frac{\partial}{\partial n} g(o(y, x)u(y)) \right] ds_y + \int_{\Omega} g(o(y, x)(-N\Delta u) d\Omega_y
\]
mit der charakteristischen Funktion

\[ c(x) = \begin{cases} 1, & x \in \Omega \\ \frac{|\Delta \varphi|}{2\pi}, & x \in \Gamma \end{cases} \]

\( g_0 \) bezeichnet die Grundlösung einer Membran für eine Einzelkraft. Die Grundlösung \( g_1 \) für Versetzung wird in dieser Einflußgleichung und auch im folgenden nicht benötigt. \( \Delta \varphi \) bezeichnet den Winkel, den der Membranrand im Aufpunkt \( x \) bildet (Siehe auch Abbildung 3).

Benötigt wird die Grundlösung der Membran in den Formen:

\[ g_0 = -\frac{1}{2\pi N} \ln r \]

sowie

\[ N \frac{\partial}{\partial n} g_0 = -\frac{1}{2\pi} \frac{r_n}{r} \]

mit

\[ r = y - x; \quad r = |r|; \quad r_n = \frac{rn}{r} \]

2.3 Zusammenhang mit der Torsion

Eine Aussage der Membrananalogie ist für die Berechnung des Torsionswiderstandsmoments entscheidend. Sie beschreibt den Zusammenhang zwischen dem Volumen unter der belasteten Membran und dem Torsionswiderstandsmoment der Fläche eines Querschnitts.

\[ I_T = 2 \int_{\Omega} u(y) d\Omega_y \]

Das zweifache Volumen unter der Membran mit der Grundfläche \( G \), die durch den Druck \( p = 2 \) belastet wird, ist gleich dem Torsionswiderstandsmoment des Querschnitts mit gleichen Abmessungen wie \( G \). Die Normalkraft \( N \), Vorspannkraft der Membran (s.o.) wird dabei auf \( N = 1 \) festgelegt. \( u(y) \) ist die Funktion für die Durchbiegung der Membran an der Stelle \( y \) (\( y \) ist ein zweidimensionaler Koordinatenvektor).

Der Rand der Grundfläche wird komplett festgehalten, und zwar auf konstantem Niveau.
2.4 Reduktion des Membranproblems

Im Hinblick auf die Aufgabenstellung – die Berechnung des Torsionswiderstandsmoments von Querschnitten – muß also ein Modell implementiert werden, das die Darstellung und Berechnung von Membranen unter Berücksichtigung folgender Besonderheiten ermöglicht:

- Dadurch, daß der Rand vollständig eingespannt ist, treten keine gemischten Randwertprobleme auf, d.h. Unbekannte auf dem Rand sind nur die Spannungen.
- Die zu berücksichtigende Belastung ist der gleichmäßige Druck \( p = 2 \). Andere Lastfälle werden für die Berechnung des Torsionswiderstandsmoments nicht benötigt.
- Aus Gründen der numerischen Darstell- und Verarbeitbarkeit wird die Kurve, die die Fläche der Membran umschreibt, durch Geradenstücke angenähert. Die Ränder stellen also geschlossene Polygonzüge dar.

Abbildung 4 zeigt ein Beispiel für das so reduzierte Membranproblem.
3 Implementierung der Randelementmethode

3.1 Übersicht über die Vorgehensweise

Der erste Schritt hin zum endgültigen Programm ist, eine polygonal berandete Membrangeometrie durch den Rechner zu erfassen. Darauf gehe ich in Abschnitt 3.2 ein.

Im zweiten Schritt werden dann die unbekannten Randgrößen (die Randspannungen) bestimmt, die Verschiebungen werden hierfür konstant gewählt. Dies stellt den Hauptaufwand des Programms dar, und wird zunächst mittels Kollokation realisiert.

Sind alle Randdaten bekannt, ist das Boundary-Element-Problem gelöst, und die Auswertung kann beginnen, insbesondere das Bestimmen des zweifachen Volumens unter der verformten Membran für das Torsionswiderstandsmoment (Dritter und letzter Schritt).

3.2 Datenstruktur zur Erfassung der Membrangeometrie


Listen, nämlich einer Ecken- und einer Kantenliste, der Fall wäre: Jeder Kante sind die begrenzenden Ecken, jeder Ecke die angreifenden Kanten direkt bekannt, ein aufwendiges Suchen in den Listen wird also vermieden.

In C++ sieht die Ringliste für den Membranrand dann stark vereinfacht folgendermaßen aus:

```cpp
class CBNode
{
    // Zeiger auf Vorgänger- und Nachfolger-Kante:
    CBEdge* pE1;
    CBEdge* pE2;
    // Datenmember: Koordinaten des Knotens:
    CVector m_pos;
    ...
};

class CBEdge
{
    // Zeiger auf Vorgänger- und Nachfolger-Knoten:
    CBNode* pN1;
    CBNode* pN2;
    ...
};

class CBModel
{
    // Zeiger auf (irgend einen) "letzten" Knoten:
    CBNode* pLastNode;
    ...
};
```

Jede Ecke CBNode enthält zwei Zeiger auf die angrenzenden Kanten CBEdge, Vorgänger und Nachfolger. CBNode trägt außerdem die geometrische Information m_pos, die Lage des Eckpunktes im zweidimensionalen Raum.


Natürlich sind zum korrekten Aufbau der Ringstruktur sowie zur Bearbeitung – wie Einfügen oder Löschen von Kanten oder Ecken – geeignete Funktionen zu implementieren. Hierbei ist darauf zu achten, daß dem Benutzer der Ring-Datenstruktur zwar ausreichende Lesezugriffsmöglichkeiten gegeben werden, aber eine Zerstörung der Datenstruktur durch unsachgemäße Handhabung verhindert wird. Hier greifen C++ Technologien wie Datenkapselung, deren Anwendung sich stark empfiehlt. Dem diesbezüglich interessierten Leser sei hier die Begleit-
Leser sei hier die *Begleit-CD* nahegelegt, die eine mögliche Implementierung der benötigten Routinen beinhaltet (Dateien *bnode.cpp*, *bedge.cpp* und *bmodel.cpp*).

Diese Ringliste stellt die grundlegende Datenstruktur für alle Boundary-Element-Berechnungen dar, sei es Kollokation, Galerkin, oder auch Berechnungen, die mehrere dieser Ringlisten einbeziehen, wie es bei der Berechnung von Querschnitten mit Löchern erforderlich wird.

### 3.3 Diskretisierung des Randes

Für beliebige Querschnitte können die Funktionen für Verschiebung und Spannung (die ja für die Anwendung der Randelementmethode bekannt sein müssen) nicht als einfache, exakte Gleichung angegeben werden, wie dies bei einfachen geometrischen Formen (z.B. Kreis oder Ellipse) der Fall ist.

Um die unbekannten Funktionen \( u(x) \) (Verschiebung) und \( t(x) \) (Spannung) auf dem Rand näherungsweise bestimmen zu können, setzt man diese Funktionen aus lokalen Ansatzfunktionen zusammen (siehe [1], Kap.3.2). Für diese Ansatzfunktionen wurde im Rahmen der Diplomarbeit ein stückweise konstanter Ansatz für die Spannung sowie ein stückweise linearer Ansatz für die Verschiebung auf dem Rand gewählt. Dies hat den Vorteil, daß bei der Spannungsfunktion die Integration unproblematisch wird. Die Verschiebungsfunktion hingegen soll keine Sprünge aufweisen; daher hier die Wahl linearer Teilfunktionen, die an den Ele-
Teilfunktionen, die an den Elementgrenzen stetig aneinander gefügt werden. Die Ordnung der Verschiebungsfunktion ist um Eins größer als die der Spannungsfunktion.

Somit gilt für $u_i(\xi)$ und $t_i(\xi)$, wobei $i$ die Elementnummer ist:

\begin{align*}
Gleichung 9 \\
_u(\xi) &= \hat{u}_i(1 - \frac{\xi}{l}) + \hat{u}_{i+1}(\xi) \\
_t(\xi) &= \hat{t}_i
\end{align*}

$\xi$ läuft über das jeweilige Element von 0 bis $l$. \textit{Abbildung 6} verdeutlicht, wie die Teilfunktionen zum Gesamtverlauf zusammengefügt werden. Sowohl Spannungs- als auch Verschiebungsfunktion sind in Ebene der Membran angetragen. (die Verläufe sind nur qualitativ; gezeigt werden soll, daß $u$ stückweise linear, $t$ stückweise konstant ist.)

\begin{figure}
\centering
\includegraphics[width=\textwidth]{image}
\caption{Abbildung 6: Teilfunktionen; Beispielverlauf}
\end{figure}

In \textit{Abbildung 6} ist der Rand in 5 Elemente und damit je 5 Ansatz-Teilfunktionen für Spannung und Verschiebung aufgeteilt. Die Teilfunktionen sind jeweils nur auf einem Element von Null verschieden (vgl. \textit{Gleichung 9}).
3.4 Schritte zur Lösung des Randwertproblems mit Hilfe der Kollokation

Wie bereits in den einleitenden Abschnitten erwähnt, ist die Hauptaufgabe beim Boundary-Element-Verfahren die Bestimmung der Randwerte. Mit Hilfe der oben beschriebenen Diskretisierung des Randes kann nun der Verlauf der Verschiebungs- und Spannungsfunktion näherungsweise bestimmt werden. Im Falle der Membrananalogie ist der Verlauf der Verschiebungsfunktion bereits komplett bekannt; er ist per Vorgabe über den gesamten Rand konstant. Unbekannt ist allerdings der Verlauf der Spannung entlang des vollständig eingespannten Randes. Es müssen also für \( n \) Elemente \( n \) Konstanten \( \hat{t} \) bestimmt werden, um die stückweise konstante Spannungsfunktion auf dem Rand zu beschreiben.

Die einfachste und gleichzeitig gebräuchlichste Methode hierfür ist das Kollokationsverfahren:

Um o.g. \( n \) Unbekannte \( \hat{t} \) eindeutig zu bestimmen, sind \( n \) voneinander unabhängige lineare Gleichungen notwendig. Man erzeugt diese Gleichungen, indem man die Einflußgleichung der Membran (Gleichung 3) in \( n \) verschiedenen Randpunkten, den Kollokationspunkten, aufstellt. Jede Gleichung enthält dann \( n \) unbekannte Konstanten \( \hat{t} \), die aus den stückweise konstanten Ansatzfunktionen stammen. Man erhält somit ein lineares Gleichungssystem, das leicht gelöst werden kann. Für dieses Gleichungssystem existiert nur eine Lösung; diese Lösung gewährleistet, daß die Einflußgleichung in allen Kollokationspunkten erfüllt wird.

Die \( n \) Kollokationspunkte müssen so gewählt werden, daß auf jedem Element ein Kollokationspunkt zu liegen kommt, um auch \( n \) verschiedene Gleichungen zu erhalten. Die Lage des Kollokationspunkts auf dem Element sollte möglichst günstig für die entsprechende Ansatzfunktion gewählt werden; sinnvollerweise bei konstanten Teilfunktionen für die Spannung \( t \) in Elementmitte, um den Fehler über das Element möglichst gering zu halten. (Die Einflußgleichung ist im Kollokationspunkt selbst exakt erfüllt; für die übrigen Punkte auf dem Randelement tritt hingegen ein Fehler auf, der auf dem Funktionsansätzen für \( u \) und \( t \) beruht, der ja den tatsächlichen Verlauf nur näherungsweise wiedergeben kann).

Aus der Wahl des Kollokationspunkts (= Aufpunkt) in Elementmitte resultiert ein angenehmer Nebeneffekt: die charakteristische Funktion \( c(x) \) ist für alle Elemente = \( \frac{1}{2} \). (vergleiche Gleichung 4).
3.4.1 Aufstellen des Gleichungssystems

Setzt man Gleichung 2, Gleichung 5 und Gleichung 6 in Gleichung 3 ein und berücksichtigt man zunächst nicht das Gebietsintegral, so erhält man für $c(x) = \frac{1}{2}$:

\[
\frac{1}{2} u(x) = \int \left[ \left( -\frac{1}{2\pi N} \ln r \right) t(y) + \frac{1}{2\pi} r \right] u(y) dy
\]

Mit oben beschriebener Diskretisierung (Abschnitt 3.3) und Aufstellung von Gleichung 10 Gleichung in $n$ Kollokationspunkten, die sich in Elementmitte befinden, erhält man ein lineares Gleichungssystem der Form

\[
H_{ij} = G_{ij}
\]

mit den Koeffizienten

\[
H_{ij} = \frac{1}{2} \delta_{ij} - \frac{1}{2\pi N} \int_{\Gamma} \left( \frac{r}{l_j} \right) d\xi_1 + \int_{\Gamma} \left( \frac{r}{l_{j-1}} \right) d\xi_2
\]

\[
G_{ij} = -\frac{1}{2\pi N} \int_{\Gamma} \ln r \, d\xi
\]

$H$ und $G$ sind $n \times n$ Matrizen; Jede Zeile $i$ repräsentiert einen Kollokationspunkt. Jede Spalte $j$ repräsentiert ein Element. Für $j - 1 = 0$ gilt $j - 1 = n$.

Es ist zu beachten, daß $r_n$ und $r$ von der Integrationsvariablen $\xi$ abhängen, da der Integrationspunkt wandert, und damit auch der Abstand $r$ zwischen Aufpunkt und Integrationspunkt variiert:

\[
r = y - x; \quad r = \left| \frac{r \, r_n}{r} \right|
\]

der Aufpunkt $x$ ist für jedes Integral konstant. $y$ läßt sich in Abhängigkeit von $\xi$ darstellen (Siehe Abbildung 7)

\[
y(\xi) = a + (b - a)\xi
\]
3.4.2 Lösen der Integrale

Wie aus Gleichung 12 und Gleichung 13 ersichtlich, sind drei verschiedene Integraltypen zu lösen. Liegt der Aufpunkt auf dem Element, über das integriert wird, lassen sich die Integrale analytisch lösen.

Für die Integrale, die in die Matrix $H$ eingingen, ist die analytische Lösung besonders einfach: $r_n$ ist in diesem Falle gleich Null, da $r$ und $n$ senkrecht aufeinander stehen, wenn der Aufpunkt auf dem Integrationselement liegt.

Ist $r_n = 0$ folgt logischerweise, daß der Ergebniswert des Integrals dann ebenfalls Null beträgt (Null integriert bleibt Null).

Für das Integral, das in die Matrix $G$ eingeht, ist die analytische Lösung ebenfalls unproblematisch, da der Integrand nur schwach singulär wird. Liegt der Aufpunkt in Elementmitte, gilt:

\[
\frac{I}{2} = \int_0^l \ln \xi \, d\xi ; \quad \Rightarrow I = l(\ln l - \ln 2 - 1).
\]

Liegt der Aufpunkt nicht auf dem Integrationselement, empfiehlt sich eine numerische Integration. Prinzipiell ist ebenfalls zwar analytische Integration möglich, aber wenig sinnvoll, da bei einer numerischen Integration mit Hilfe der Gauss-Legendre-Polynome bei 4 bis 8 Stützstellen hohe Genauigkeit erreicht wird, und die analytische Bestimmung hier aufgrund zahlreicher zu berechnender arctan- und $\ln$- Funktionen der Stammfunktionen die benötigte Rechenzeit ungünstig beeinflußt.
3.4.3 Lösen des Gleichungssystems

Sind die Matrizen \( H \) und \( G \) bestimmt, kann das Gleichungssystem nach \( \hat{t} \) aufgelöst werden. Da das Gleichungssystem nicht symmetrisch ist, muß die komplette Matrix verwendet werden. Ich konnte auf den pivotisierenden Gleichungslöser von Herrn Prof. Holzer zurückgreifen. Der Gleichungslöser verarbeitet Gleichungen der Form \( G\hat{t} = r \); die rechte Seite \( r \) kann vorher direkt aus \( Hu \) berechnet werden.
3.5 Programmierung der Kollokation in C++

Als Datengrundlage dient die oben beschriebene Klasse \textit{CBModel}. Die Kollokationsklasse habe ich \textit{CColloc} genannt. \textit{CColloc} wird durch den Konstruktoraufruf mit einer Instanz \textit{CBModel} per Zeiger verbunden.

Ein kurzer Auszug aus den entsprechenden Headerdateien:

```cpp
#include "element.h"

class CGeneralColloc
{
    // Hier wird die Anzahl Elemente festgehalten:
    int Size;
    // Zeiger auf die Matrizen und Vektoren; erhalten später Dimension Size.
    matrix *H, *G;
    double *t, *u, *p;
    // Funktion für die aufgebrachte Randverschiebung:
    CFunction2d* pf;
    void SetFunction(CFunction2d* npf) { pf=npf; };
    ...
};

class CColloc : public CGeneralColloc
{
    // Zeiger auf das Modell der Randdaten:
    CBModel* pBM;
    // zentrale Funktionen:
    virtual void ComputeHGP(int ng);
    virtual void Compute();
    ...
};
```

Der C++-fime Leser fragt sich wahrscheinlich, warum \textit{CColloc} von \textit{CGeneralColloc} abgeleitet wurde. Dies hat programmiertechnische Gründe, um später mit sehr geringem Aufwand eine Implementierung des Galerkin-Verfahrens zu ermöglichen. Aus demselben Grund sind übrigens auch \textit{ComputeHGP} und \textit{Compute} virtuell... dazu später mehr.

\textit{CColloc} enthält als zentrale, zunächst wichtige Funktionen

\textit{ComputeIntegralsHGP}, sowie

\textit{Compute}.

Wichtige \textit{Datenmember} der Klasse sind die Matrizen \textit{G} und \textit{H}, sowie die Vektoren \textit{t} und \textit{u}, die mit den hier gewählten Bezeichnungen übereinstimmen (bis auf \textit{t} für \( \hat{\mathbf{t}} \), \textit{u} für \( \hat{\mathbf{u}} \)). Je nach Anzahl der Elemente, die die Ringliste in \textit{CBModel} enthält, werden Matrizen und Vektoren in der richtigen Dimension initialisiert.
Compute setzt zunächst den Vektor $u$, bringt also eine bekannte Randverschiebung auf. Hier können beliebige Funktionen für die Verschiebungsfunktion $u$ angesetzt werden, was später für Testzwecke benötigt wird. Dann berechnet ComputeHGP die Matrizen $H$ und $G$. Dazu wird die Ringliste in zwei ineinander geschachtelten Schleifen durchlaufen. Die äußere Schleife setzt den Kollokationspunkt, der stets in Elementmitte liegt, um ein Element weiter. Die innere Schleife setzt das aktuelle Integrationselement weiter. Diese wichtige Funktion habe ich unter Kapitel 7.1: Realisierung des Galerkin-Verfahrens mit Kommentaren für Vergleichszwecke aufgelistet; bei Interesse bitte vorblättern!


Die Funktion Compute stellt sich in C++ , auf die wichtigsten Programmteile beschränkt, folgendermaßen dar:

```c++
void CColloc::Compute()
{
  ...
  // Matrizen auf richtige Größe initialisieren:
  Reset();
  ...
  // Berechnung der Matrizen $H$ und $G$, sowie Vektor $p$ (=Lastvektor; wird
  // zunächst noch nicht benötigt.
  ComputeHGP(GaussPoints);
  // Verschiebungen und konjugierte Soll-Spannungen mit Funktion in allen
  // Kollokationspunkten berechnen:
  ComputeVectorsUT();
  // rechte Seite ausrechnen: $rs = H*u - p$
  double *rs=new double[Size];
  for (i=1;i<=Size;i++)
  {
    rs[i-1]=-p[i-1];
    for (j=1;j<=Size;j++)
      rs[i-1]+=(*H)(i,j)*u[j-1];
  }
  // LGS nach $t$ lösen: ( Form : $G*t=rs$ )
  SolveScaledRowGauss(*G,rs,t,Size);
  // aufräumen:
  delete[] rs;
}
```
3.6 Möglichkeiten zum Testen der Ergebnisse und Auswertung

Die beste Möglichkeit, um die erhaltenen $\hat{t}$ zu überprüfen, die ja direkt die Spannungen in den Kollokationspunkten angeben, ist folgende:
Als aufgebrachtes $u$ wird eine Funktion $u$ in Abhängigkeit von $x$ ($x_1, x_2$) angesetzt. Ist $\Delta u(x) = 0$, genügt die Funktion der Membrangleichung *Gleichung 1* für $p = 0$, also genau dem hier betrachteten Fall, daß zunächst kein Druck auf der Membran lastet.
Die Werte $\hat{u}$ sind also bekannt. Mit *Gleichung 2* können nun exakte Soll-Werte für die Spannungen bestimmt werden, die dann mit den mittels Kollokation bestimmten $\hat{t}$ verglichen werden können.

Ich habe zunächst ein Rechteck $ABCD$, $AB = 2 \ BC$, unterteilt in 6 Elemente, untersucht. Als Funktion $u$ habe ich

$$u(x_1, x_2) = x_1^2 - x_2^2$$

gewählt. Wie leicht überprüft werden kann wird *Gleichung 1* erfüllt, da $\Delta u(x) = 0$ für alle $x$ erfüllt wird.

Im Programmcode können Funktionen diesen oder anderen Typs mit der Funktion *SetFunction* dem Kollokationsmodell zugeordnet werden. Dazu muß eine neue Funktionenklasse von einer zweidimensionalen Ur-Klasse abgeleitet werden, um korrekt von der Klasse *CColloc* verarbeitet werden zu können.

Für die obige Funktion sieht dies im Quelltext so aus:

```cpp
class CFunction2d
// Definition der Ur-Funktion (definiert in func2d.h)
{
    public:
    // Funktionswert an Stelle x,y zurückgeben
    virtual double operator() (CVec x)=0;
    // Wert der 1. Ableitung nach x an Stelle x,y zurückgeben
    virtual double x(CVec x)=0;
    // Wert der 1. Ableitung nach y an Stelle x,y zurückgeben
    virtual double y(CVec x)=0;
    // Laplace-Operator auf f in x,y anwenden
    virtual double Laplace(CVec x)=0;
};
class Cf2 : public CFunction2d
// Definiert Funktion des Typs f(x,y) = ax^2 + by^2
{
    double a,b;
    public:
```
Hier nun das Ergebnis des Tests:

Abbildung 9: Diskretisierung I des Rechtecks

<table>
<thead>
<tr>
<th>El.Nr.</th>
<th>u[x]</th>
<th>t[x] exakt</th>
<th>t[x] kollok.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,05</td>
<td>0</td>
<td>0,0794617</td>
</tr>
<tr>
<td>1</td>
<td>0,45</td>
<td>0</td>
<td>0,132151</td>
</tr>
<tr>
<td>2</td>
<td>0,75</td>
<td>1,78885</td>
<td>1,91762</td>
</tr>
<tr>
<td>3</td>
<td>0,25</td>
<td>-0,894427</td>
<td>-0,958733</td>
</tr>
<tr>
<td>4</td>
<td>-0,15</td>
<td>-0,894427</td>
<td>-1,01756</td>
</tr>
<tr>
<td>5</td>
<td>-0,05</td>
<td>0</td>
<td>-0,0712201</td>
</tr>
</tbody>
</table>

Mittlerer abs. Fehler \( \frac{\sum(t_a-t_k)}{n} \): 0,0998399
Maximaler Fehler: 0,132151

Erstaunlich, wie bereits bei äußerst grober Diskretisierung des Randes relativ gute Ergebnisse erzielt werden. Zur Erinnerung: Die gewählten Spannungsfunktionen sind konstant über jedem Element, die Verschiebungsfunktionen linear!
Um das Ergebnis zu verbessern, kann nun die Elementzahl erhöht werden, z.B. auf 72 Elemente.

Abbildung 10: Diskretisierung II des Rechtecks

Bei dieser Unterteilung des Randes erhält man auszugsweise:

Tabelle 2

<table>
<thead>
<tr>
<th>El.Nr.</th>
<th>u(x)</th>
<th>t(x) exakt</th>
<th>t(x) kollok.</th>
<th>abs. Fehler</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,000347222</td>
<td>0</td>
<td>0,00540085</td>
<td>0,00540085</td>
</tr>
<tr>
<td>6</td>
<td>0,0586806</td>
<td>0</td>
<td>0,000572633</td>
<td>0,000572633</td>
</tr>
<tr>
<td>13</td>
<td>0,253125</td>
<td>0</td>
<td>0,000746635</td>
<td>0,000746635</td>
</tr>
<tr>
<td>21</td>
<td>0,642014</td>
<td>0</td>
<td>-0,00252002</td>
<td>0,00252002</td>
</tr>
<tr>
<td>22</td>
<td>0,703125</td>
<td>0</td>
<td>-0,00588073</td>
<td>0,00588073</td>
</tr>
<tr>
<td>23</td>
<td>0,767014</td>
<td>0</td>
<td>-0,0710211</td>
<td>0,0710211</td>
</tr>
<tr>
<td>24</td>
<td>0,799653</td>
<td>1,78885</td>
<td>1,92932</td>
<td>0,140462</td>
</tr>
<tr>
<td>25</td>
<td>0,796875</td>
<td>1,78885</td>
<td>1,76225</td>
<td>0,0266084</td>
</tr>
<tr>
<td>26</td>
<td>0,791319</td>
<td>1,78885</td>
<td>1,78248</td>
<td>0,00637319</td>
</tr>
<tr>
<td>40</td>
<td>0,328125</td>
<td>-0,894427</td>
<td>-0,896412</td>
<td>0,00198513</td>
</tr>
<tr>
<td>70</td>
<td>-0,003125</td>
<td>0</td>
<td>-0,000254656</td>
<td>0,000254656</td>
</tr>
</tbody>
</table>

Mittlerer absoluter Fehler: 0.011387

Man kann hier deutlich sehen, wie die Methode der Randelemente zu den Ecken hin an ihre Grenzen (zwischen Element 23 und 24 befindet sich die Ecke) stößt. Da der Fehler jedoch räumlich immer stärker begrenzt wird (die Elemente werden kleiner), wird die Lösung insgesamt betrachtet jedoch auch in den Ecken besser.
Eine aussagekräftige Bewertung der Randspannungsergebnisse \( \hat{t} \) kann über eine Integration der Fehlerquadrate über den Rand erfolgen. Dazu wird der Fehler \( t_{exakt} - t_{kollok} \) in allen für die Gauss-Quadratur benötigten Integrationspunkten bestimmt. Da die Spannungsfunktion \( t \), die aus der Kollokation gewonnen wird, stückweise konstant ist, ist die Berechnung des gesamten Fehlers \( TF \) einfach.

\[
TF = \sqrt{\sum_{i=1}^{n} \left( t(\xi) - \hat{t}(\xi) \right)^2} d\xi
\]

Im doppelt-logarithmischen Maßstab aufgetragen kann man erkennen, wie der Fehler \( TF \) bei wachsender Elementzahl stetig gegen Null konvergiert.

Tabelle 3

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Fehler TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0,17014</td>
</tr>
<tr>
<td>12</td>
<td>0,101553</td>
</tr>
<tr>
<td>18</td>
<td>0,0993131</td>
</tr>
<tr>
<td>24</td>
<td>0,0911695</td>
</tr>
<tr>
<td>36</td>
<td>0,0795224</td>
</tr>
<tr>
<td>48</td>
<td>0,0711551</td>
</tr>
<tr>
<td>72</td>
<td>0,0600019</td>
</tr>
<tr>
<td>108</td>
<td>0,0500396</td>
</tr>
<tr>
<td>156</td>
<td>0,0421749</td>
</tr>
<tr>
<td>234</td>
<td>0,0347667</td>
</tr>
<tr>
<td>348</td>
<td>0,0286871</td>
</tr>
</tbody>
</table>

Abbildung 11: Konvergenzkurve Fehler TF für Funktion 1
3.7 Aufbringen der Gebietslast: gleichmäßiger Druck

Das Gebietsintegral in Gleichung 3 wurde bis jetzt nicht berücksichtigt. Für die Berechnung des Torsionswiderstandsmoments muß die Membran jedoch mit einer Gebietslast, dem Druck \( p = 2 \), belastet werden.

Hier die Formel für das Gebietsintegral mit Gleichung 1 und Gleichung 6:

\[
\int_{\Omega} g(y, x)(-N\Delta u)d\Omega = p \int_{\Omega} -\frac{1}{2\pi N} \ln r d\Omega
\]

Dieses Gebietsintegral kann in ein Randintegral umgeformt werden (siehe [1], S. 126):

\[
p \int_{\Omega} -\frac{1}{2\pi N} \ln r d\Omega = \frac{p}{8\pi N} \int_{\Gamma} r(2\ln r - 1) r d\Gamma
\]

Das zu lösende Gleichungssystem erweitert sich zu

\[
\mathbf{H} u = \mathbf{G} \mathbf{i} + \mathbf{\hat{p}}
\]

Der gleichmäßige Druck hat gemäß Gleichung 3 Einfluß auf jeden Kollokationspunkt, daher ein Vektor \( \mathbf{\hat{p}} \).

\( \mathbf{\hat{p}} \) hat die Koeffizienten:

\[
\mathbf{\hat{p}} = \frac{p}{8\pi N} \sum_{j=1}^{n} r(2\ln r - 1) r_{n} d\Sigma
\]

Die laut Gleichung 7 in \( r \) und \( r_{n} \) enthaltene zweidimensionale Variable \( y \) (Integrationspunkt), kann wie oben (siehe 3.4.1 und Abbildung 7) beschrieben in Abhängigkeit einer Integrationsvariable \( \xi \) dargestellt werden. Der Aufpunkt = Kollokationspunkt \( x \) ist für jeden Koeffizienten \( \mathbf{\hat{p}} \) natürlich wieder konstant.

Die Lösung des Gleichungssystems ist analog zu 3.4.3, mit der rechten Seite \( \mathbf{H} u - \mathbf{\hat{p}} \).

Die korrekte Implementierung der Gebietslast kann wie unter Abschnitt 3.6 mit Hilfe einer aufgebrachten Verschiebungsfunktion überprüft werden.

Wird der ein Druck von \( p = 2 \) aufgebracht, ist z. B.

\[
u(x_1, x_2) = -\frac{1}{2} x_1^2 - \frac{1}{2} x_2^2
\]

eine gültige Funktion, da Gleichung 1 erfüllt wird (für \( N = 1 \)): \( \Delta u = -2 \).
Mit Rechteck ABCD aus dem vorigen Beispiel stellt sich folgender Verlauf von TF ein, wenn die Elementeinteilung gleichmäßig verfeinert wird:

Tabelle 4

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Fehler TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0,0726046</td>
</tr>
<tr>
<td>12</td>
<td>0,0455837</td>
</tr>
<tr>
<td>18</td>
<td>0,0423264</td>
</tr>
<tr>
<td>24</td>
<td>0,0370863</td>
</tr>
<tr>
<td>36</td>
<td>0,0308616</td>
</tr>
<tr>
<td>48</td>
<td>0,0269822</td>
</tr>
<tr>
<td>72</td>
<td>0,0222538</td>
</tr>
<tr>
<td>108</td>
<td>0,0183018</td>
</tr>
<tr>
<td>156</td>
<td>0,0153</td>
</tr>
<tr>
<td>234</td>
<td>0,0125388</td>
</tr>
<tr>
<td>348</td>
<td>0,0103078</td>
</tr>
</tbody>
</table>

Abbildung 12: Konvergenzkurve Fehler TF für Funktion 2

Zum Vergleich ist Verlauf 1 aus vorigem Beispiel nochmals im Diagramm dargestellt. Die Kurven verlaufen fast parallel, der betragsmäßig kleinere Fehler bei Verlauf 2 liegt daran, daß die Funktion u kleinere Verschiebungen aufbringt (Faktor ½), und damit auch die Spannungen bzw. der Spannungsfehler betragsmäßig kleiner werden.
4 Berechnung des Torsionswiderstandsmoments

Nach der Membrananalogie gilt:

\[ I_T = 2 \int u(y) d\Omega_y \]

Um die Idee der Randintegralmethode konsequent weiterzuführen, ist es zweckmäßig, dieses Gebietsintegral in ein Randintegral umzuformen. Dies wird mit der Greenschen Integralgleichung für den ebenen Fall möglich ([4], S. 350):

\[ \int_{\Omega} (u \Delta h - h \Delta u) d\Omega = \int_{\Gamma} (u \frac{\partial h}{\partial n} - h \frac{\partial u}{\partial n}) d\Gamma \]

\( h \) ist eine beliebige, in ihren 2. partiellen Ableitungen stetige Hilfsfunktion. Wählt man \( h \) so, daß \( \Delta h = 1 \) ist, gilt unter Berücksichtigung von Gleichung 1 und Gleichung 2 für \( N = 1 \):

\[ \frac{1}{2} I_T = \int_{\Omega} u(y) d\Omega_y = \int_{\Gamma} \frac{\partial h}{\partial n} (y) d\Gamma_y - \int_{\Gamma} h(y) t(y) d\Gamma_y + (-p) \int_{\Omega} h(y) d\Omega_y \]

Bei Querschnitten ohne Löcher kann die konstante Randverschiebungsfunktion \( u \) frei gewählt werden. Wählt man hier \( u(y) = 0 \), wird das Integral \( I_1 \) zu Null.

Das Integral \( I_2 \) kann leicht analytisch bestimmt werden, da für die Spannungsfunktion ein stückweiser konstanter Ansatz gewählt wurde. \( t(y) \) kann also vor das Integral gezogen werden, wertet man das Integral elementweise aus.

Das verbleibende Gebietsintegral \( I_3 \) kann man, je nach Wahl der Hilfsfunktion \( h \), ebenfalls in ein Randintegral umwandeln, und analytisch bestimmen.

Ich habe zwei verschiedene Hilfsfunktionen \( h \) ausprobiert; und zwar

\[ h(y) = \frac{1}{2} y_1^2 \quad \text{und} \quad h(y) = \frac{1}{4} (y_1^2 + y_2^2) \]

Diese werden im Folgenden als Funktion 1 und Funktion 2 bezeichnet.

Für die erste Funktion wird \( I_3 \) zu:

\[ I_3 = \int_{\Gamma} \frac{1}{3} y_1^3 m \ d\Gamma_y \]

Für die zweite Funktion kann \( I_3 \) analog bestimmt werden.

Obwohl die Wahl der Hilfsfunktionen eigentlich beliebig ist, und nach Green keine Auswirkung auf das zu berechnende Integral haben sollte, liefern verschiedene Funktionen dennoch unterschiedliche Ergebnisse. Dies kann folgendermaßen erklärt werden: Im Integral \( I_2 \) wurde die Normalableitung der Verschiebungsfunktion \( \frac{\partial u}{\partial n} \) unter Anwendung von **Gleichung 2** durch die Spannungsfunktion \( \sigma \) ersetzt, um direkt die berechneten Randspannungswerte verwenden zu können. Diese sind aber nur Näherungslösungen (und zwar stückweise konstante Funktionen), weisen also Fehler auf. Da die fehlerhafte Funktion dann mit unterschiedlichen Hilfsfunktionen multipliziert wird, erhält man für das Integral \( I_2 \) natürlich ebenfalls differierende Ergebnisse.

Es läßt sich schwerlich sagen, welche der beiden Funktionen die besseren Ergebnisse liefert. Wie die nachfolgenden Beispiele zeigen, ist **Funktion 1** beim Dreiecksquerschnitt besser, **Funktion 2** dagegen beim I-Profil. Da aber die Funktionswahl keinen Einfluß auf die Konvergenzgeschwindigkeit hat (die Kurven verlaufen gleich steil im doppelt-logarithmischen Maßstab), wird der Unterschied bei feinerer Diskretisierung des Netzes auch immer weniger relevant. Somit würde ich **Funktion 1** vorziehen, da sie den kleinstmöglichen Rechenaufwand benötigt (ungefähr die halbe Zeit wie **Funktion 2**).

**Zur Organisation im Programm:**

Die Berechnung des Torsionswiderstandsmoments übernimmt die Funktion \( \text{ComputeTorque} \) der Klasse \( \text{CColloc} \). \( \text{ComputeTorque} \) berechnet das gesamte \( I_T \), indem die über die Hilfsklasse \( \text{CCollocElement} \) berechneten Anteile jedes Elementes aufaddiert werden.

In den folgenden Tabellen und Abbildungen sind die Ergebnisse für einige Beispielquerschnitte dargestellt, jeweils mit **Funktion 1** und **Funktion 2** berechnet. Der angegebene Fehler bezieht sich auf Referenzwerte, die mit Hilfe eines FE-Programms von Herrn Prof. Holzer und anschließender Extrapolation berechnet wurden.
Tabelle 5: Rechteck

<table>
<thead>
<tr>
<th>Elemente</th>
<th>$I_T$ mit $F_1$</th>
<th>Fehler</th>
<th>$I_T$ mit $F_2$</th>
<th>Fehler</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0,0570574</td>
<td>0,0004161374</td>
<td>0,0459377</td>
<td>0,00012374</td>
</tr>
<tr>
<td>12</td>
<td>0,0374444</td>
<td>0,0003977671</td>
<td>0,034518</td>
<td>0,00027866</td>
</tr>
<tr>
<td>18</td>
<td>0,0323539</td>
<td>0,0003388471</td>
<td>0,0311167</td>
<td>0,00078335</td>
</tr>
<tr>
<td>24</td>
<td>0,0305902</td>
<td>0,000181612</td>
<td>0,029932</td>
<td>0,00029292</td>
</tr>
<tr>
<td>36</td>
<td>0,0293658</td>
<td>0,0002080564</td>
<td>0,0291224</td>
<td>0,00083571</td>
</tr>
<tr>
<td>48</td>
<td>0,0289976</td>
<td>0,0001466693</td>
<td>0,028861</td>
<td>0,00096907</td>
</tr>
<tr>
<td>72</td>
<td>0,0287444</td>
<td>0,0000561153</td>
<td>0,0286914</td>
<td>0,00037573</td>
</tr>
<tr>
<td>108</td>
<td>0,0286102</td>
<td>0,0000216025</td>
<td>0,0286253</td>
<td>0,00041586</td>
</tr>
<tr>
<td>156</td>
<td>0,0285946</td>
<td>0,000037084</td>
<td>0,0285914</td>
<td>0,00025889</td>
</tr>
<tr>
<td>234</td>
<td>0,0285888</td>
<td>0,000016793</td>
<td>0,0285876</td>
<td>0,00012594</td>
</tr>
</tbody>
</table>

Referenzwert für $I_T$: 0,285584

Tabelle 6: Dreieck

<table>
<thead>
<tr>
<th>Elemente</th>
<th>$I_T$ mit $F_1$</th>
<th>Fehler</th>
<th>$I_T$ mit $F_2$</th>
<th>Fehler</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>876228</td>
<td>2,1279439</td>
<td>774084</td>
<td>1,75876275</td>
</tr>
<tr>
<td>6</td>
<td>613843</td>
<td>1,18767886</td>
<td>577295</td>
<td>1,05742522</td>
</tr>
<tr>
<td>9</td>
<td>409717</td>
<td>0,46019295</td>
<td>395374</td>
<td>0,4097584</td>
</tr>
<tr>
<td>12</td>
<td>345127</td>
<td>0,23000025</td>
<td>337840</td>
<td>0,20403007</td>
</tr>
<tr>
<td>18</td>
<td>304267</td>
<td>0,08437904</td>
<td>301532</td>
<td>0,07463176</td>
</tr>
<tr>
<td>24</td>
<td>292094</td>
<td>0,04095691</td>
<td>290746</td>
<td>0,03619147</td>
</tr>
<tr>
<td>36</td>
<td>284712</td>
<td>0,01468686</td>
<td>284221</td>
<td>0,01293698</td>
</tr>
<tr>
<td>54</td>
<td>282038</td>
<td>0,00522825</td>
<td>281881</td>
<td>0,00439744</td>
</tr>
<tr>
<td>78</td>
<td>281164</td>
<td>0,00204212</td>
<td>281094</td>
<td>0,00179264</td>
</tr>
<tr>
<td>117</td>
<td>280794</td>
<td>0,00072347</td>
<td>280769</td>
<td>0,00063438</td>
</tr>
<tr>
<td>174</td>
<td>280665</td>
<td>0,00026373</td>
<td>280656</td>
<td>0,00023165</td>
</tr>
<tr>
<td>261</td>
<td>280618</td>
<td>9,62258-05</td>
<td>280615</td>
<td>8,55348-05</td>
</tr>
<tr>
<td>390</td>
<td>280601</td>
<td>3,56398-05</td>
<td>280600</td>
<td>3,20758-05</td>
</tr>
</tbody>
</table>

Referenzwert für $I_T$: 280591

Tabelle 7: T-Profil

<table>
<thead>
<tr>
<th>Elemente</th>
<th>$I_T$ mit $F_1$</th>
<th>Fehler</th>
<th>$I_T$ mit $F_2$</th>
<th>Fehler</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>195298</td>
<td>1,07414755</td>
<td>230439</td>
<td>1,44735987</td>
</tr>
<tr>
<td>11</td>
<td>176631</td>
<td>0,8758961</td>
<td>171321</td>
<td>0,81950165</td>
</tr>
<tr>
<td>22</td>
<td>119289</td>
<td>0,26689975</td>
<td>119070</td>
<td>0,26453788</td>
</tr>
<tr>
<td>44</td>
<td>100548</td>
<td>0,06786238</td>
<td>100508</td>
<td>0,06743757</td>
</tr>
<tr>
<td>88</td>
<td>95736,6</td>
<td>0,01676328</td>
<td>95735,9</td>
<td>0,01675584</td>
</tr>
<tr>
<td>176</td>
<td>94576,1</td>
<td>0,00443828</td>
<td>94577,6</td>
<td>0,00445421</td>
</tr>
<tr>
<td>352</td>
<td>94285,4</td>
<td>0,00135092</td>
<td>94285,9</td>
<td>0,00135623</td>
</tr>
</tbody>
</table>

Referenzwert für $I_T$: 94158,2

Tabelle 8: I-Profil

<table>
<thead>
<tr>
<th>Elemente</th>
<th>$I_T$ mit $F_1$</th>
<th>Fehler</th>
<th>$I_T$ mit $F_2$</th>
<th>Fehler</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>803,572</td>
<td>12,5381308</td>
<td>606,12</td>
<td>9,21157019</td>
</tr>
<tr>
<td>16</td>
<td>317,041</td>
<td>4,34132913</td>
<td>463,326</td>
<td>6,80585684</td>
</tr>
<tr>
<td>28</td>
<td>257,926</td>
<td>3,34539273</td>
<td>334,375</td>
<td>4,63363265</td>
</tr>
<tr>
<td>52</td>
<td>201,449</td>
<td>2,39389988</td>
<td>245,805</td>
<td>3,1418491</td>
</tr>
<tr>
<td>100</td>
<td>146,845</td>
<td>1,47396228</td>
<td>172,596</td>
<td>1,9078007</td>
</tr>
<tr>
<td>196</td>
<td>107,939</td>
<td>0,81849579</td>
<td>122,32</td>
<td>1,06077882</td>
</tr>
<tr>
<td>200</td>
<td>87,1581</td>
<td>0,46839083</td>
<td>95,1128</td>
<td>0,60240716</td>
</tr>
<tr>
<td>392</td>
<td>72,7578</td>
<td>0,22578265</td>
<td>76,5684</td>
<td>0,2899815</td>
</tr>
</tbody>
</table>

Referenzwert für $I_T$: 59,3562
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 13: Geometrie Dreieck

Abbildung 14: Geometrie Rechteck

Abbildung 15: Verlauf des Fehlers gegen Referenzwert Rechteck und Dreieck
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 16: Geometrie T-Profil

Abbildung 17: Geometrie I-Profil

Abbildung 18: Verlauf des Fehlers gegen Referenzwert T- und I-Profil
Auf Abbildung 18 wird deutlich, daß die Konvergenzgeschwindigkeit stark von der Geometrie des Querschnitts abhängt. So erweisen sich dünnwandige Querschnitte, sowie Querschnitte mit einspringenden Ecken als besonders ungünstig.


In Abschnitt 5.2 möchte ich daher auf Möglichkeiten eingehen, wie Fehlergrößen nicht im gesamten, sondern für die einzelnen Elemente getrennt ermittelt werden können. Mit Hilfe dieser Fehlergrößen kann die Randaufteilung dann adaptiv erfolgen.
5 Fehlergrößen

Bisher habe ich die Ergebnisse stets nach demselben Muster bewertet: Entweder wurden Randverschiebungen aufgebracht, für die die Spannungsverteilung bereits bekannt war, und somit leicht eine Beurteilung der ermittelten Spannungswerte erfolgen konnte; oder die mit der BE-Methode ermittelten Torsionsträgheitsmomente wurden mit bekannten, mittels FE-Methode bestimmten Werten verglichen.

Dies ist natürlich äußerst unbefriedigend. Der Anwender möchte möglichst direkt Ergebnisse erhalten, ohne sich darum kümmern zu müssen, ob eine Genauigkeit erreicht wurde, die seinen Ansprüchen genügt. „Schön“ wäre es für den Anwender, wenn er nur eine Angabe einer angestrebten Genauigkeit des Ergebnisses (z.B. Fehler unter 1%) vornehmen müßte, um dann innerhalb kürzest möglich der Bearbeitungszeit ein Endergebnis zu erhalten, das dieser Vorgabe entspräche.


Ich möchte hierzu zwei sinnvolle Möglichkeiten vorstellen:

Die Methode des Kräftegleichgewichts, sowie die Methode der elementweisen Bestimmung des $u^2$-Residuums.

Beide Methoden kombiniert stellen ein mächtiges Werkzeug zur Beurteilung der Ergebnisse dar. Durch das elementweise Messen von Fehlergrößen wird auch eine adaptive Verfeinerung der Randaufteilung ermöglicht.
5.1 Fehler bezüglich des Kräftegleichgewichts

Ein wichtiger Unterschied in der Methode der Randelemente im Vergleich zu den finiten Elementen besteht darin, daß bei den finiten Elementen das Kräftegleichgewicht exakt erfüllt wird. Dies ist bei der Randelementmethode nicht der Fall.

Das Ergebnis kann also bewertet werden, indem das Kräftegleichgewicht überprüft wird. Je besser diese Bedingung erfüllt wird, desto besser wird auch das Ergebnis für das Torsionswiderstandsmoment.

Das Kräftegleichgewicht für die gesamte Membran ist leicht aufzustellen: Die Summe der Randeinspannungskräfte muß gleich dem gesamten aufgebrachten Druck sein. In einer Gleichung dargestellt sieht das folgendermaßen aus:

\[
\int_{\Omega} p d\Omega = \int_{\Gamma} f(x) d\Gamma_x
\]

Das Gebietsintegral kann, analog zum Integral \( I_3 \) in Gleichung 20, in ein Randintegral umgeformt werden und analytisch bestimmt werden. Das Randintegral ist ebenfalls unproblematisch analytisch lösbar, da die Spannungsfunktion \( f \), wie inzwischen mehrfach erwähnt, elementweise konstant ist.

Der aufgebrachte Druck ist bekannt und ein fixer Wert \( p = 2 \). Wird die berechnete Summe der Randspannungen auf diesen Druck bezogen, erhält man eine Fehlergröße, die ich als \( T_R \) bezeichnen möchte:

\[
T_R = \frac{\int_{\Gamma} f(x) d\Gamma_x}{\int_{\Omega} p d\Omega} \sqrt{\int_{\Omega} p d\Omega - 1}
\]

Nachfolgend habe ich den Fehler \( T_R \) in gewohnter Form, d.h. im doppelt-logarithmischen Maßstab für die bereits bekannten vier Beispielquerschnitte Rechteck, Dreieck, T- und I-Profil dargestellt (Abbildung 19). Datengrundlage für das Schaubild sind Tabelle 9 bis Tabelle 12.
Tabelle 9: Rechteck

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Fehler TR</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0,0355955</td>
</tr>
<tr>
<td>12</td>
<td>0,0107873</td>
</tr>
<tr>
<td>24</td>
<td>0,00245828</td>
</tr>
<tr>
<td>48</td>
<td>0,00051399</td>
</tr>
<tr>
<td>96</td>
<td>0,00010159</td>
</tr>
<tr>
<td>192</td>
<td>1,93E-05</td>
</tr>
<tr>
<td>384</td>
<td>3,58E-06</td>
</tr>
</tbody>
</table>

Tabelle 10: Dreieck

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Fehler TR</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0,109209</td>
</tr>
<tr>
<td>6</td>
<td>0,0390751</td>
</tr>
<tr>
<td>12</td>
<td>0,00779144</td>
</tr>
<tr>
<td>24</td>
<td>0,00144105</td>
</tr>
<tr>
<td>48</td>
<td>0,00025485</td>
</tr>
<tr>
<td>96</td>
<td>4,40E-05</td>
</tr>
<tr>
<td>192</td>
<td>7,48E-06</td>
</tr>
<tr>
<td>384</td>
<td>1,26E-06</td>
</tr>
</tbody>
</table>

Tabelle 11: T-Profil

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Fehler TR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0,0262414</td>
</tr>
<tr>
<td>13</td>
<td>0,0120736</td>
</tr>
<tr>
<td>26</td>
<td>0,00347423</td>
</tr>
<tr>
<td>44</td>
<td>0,00105361</td>
</tr>
<tr>
<td>88</td>
<td>0,0002226</td>
</tr>
<tr>
<td>104</td>
<td>1,63E-04</td>
</tr>
<tr>
<td>176</td>
<td>4,43E-05</td>
</tr>
<tr>
<td>208</td>
<td>3,21E-05</td>
</tr>
<tr>
<td>352</td>
<td>8,49E-06</td>
</tr>
</tbody>
</table>

Tabelle 12: I-Profil

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Fehler TR</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0,00534246</td>
</tr>
<tr>
<td>16</td>
<td>0,0017846</td>
</tr>
<tr>
<td>28</td>
<td>0,00184976</td>
</tr>
<tr>
<td>52</td>
<td>0,00107848</td>
</tr>
<tr>
<td>100</td>
<td>0,00066385</td>
</tr>
<tr>
<td>200</td>
<td>0,00021012</td>
</tr>
<tr>
<td>400</td>
<td>4,87E-05</td>
</tr>
<tr>
<td>800</td>
<td>1,03E-05</td>
</tr>
</tbody>
</table>

**Fazit:**

Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 19: Fehler $T_R$

Elemente

Fehler $T_R$

Rechteck
Dreieck
T-Profil
I-Profil
5.2 Fehler bezüglich der Verschiebung: $u^2$ - Residuum

Der Fehler $\bar{T}_R$ kann zwar generelle Aussagen über die Qualität der Endlösung treffen. Unbeantwortet bleibt jedoch die Frage, wie sich der Fehler über die einzelnen Elemente verteilt. Um diese Frage zu klären, kann der das Residuum der Verschiebung $r_{u^2}$ an jeder beliebigen Stelle des Randes berechnet werden.

Die Vorgehensweise hierbei ist folgendermaßen:
Man berechnet für den Randpunkt, für den der Verschiebungsfehler ermittelt werden soll, die Verschiebung $u(x)$ mit der Einflußgleichung (Gleichung 10):

$$\frac{1}{2} u(x) = \int_{\Gamma} \left( - \frac{1}{2\pi N} \ln r \right) t(y) + \left( \frac{1}{2\pi} r_n \right) u(y) \, ds_y$$

Hier gehen sowohl sämtliche angesetzte Teil-Verschiebungsfunktionen (die vorgegeben wurden, im Spezialfall Torsion ohne Löcher konstant gleich Null), als auch sämtliche angesetzte Teil-Spannungsfunktionen (die für jedes einzelne Element durch Kollokation bestimmt wurden), ein, da über den gesamten Rand integriert wird.

Die wie oben mit der Einflußgleichung berechneten Werte $u_{ber}(x)$ können dann mit den durch die Teilfunktionen vorgegebenen $u_{vorg}(x)$ verglichen werden; im Falle des Torsionsproblems bei Querschnitten ohne Löcher gilt $u_{vorg}(x) = 0$.

Für das Residuum $r_{u^2}$ im einzelnen Punkt gilt also, wenn man noch quadriert, um durchweg positive Werte zu erhalten und „Ausreißer“ stärker zu berücksichtigen:

Gleichung 21: $r_{u^2}(x) = (u_{ber}(x) - u_{vorg}(x))^2$

Die Residuen, nicht quadriert (also $u_{ber} - u_{vorg}$), für die Querschnitte I-Profil und Dreieck unter dem Druck $p = 2$ sind in den folgenden Abbildungen stark überhöht dargestellt (Jeweils gleiche Überhöhung in den Abbildungen bei gleichem Querschnitt). Hier wird deutlich: In den Kollokationspunkten, die die Mittelpunkte der Elemente darstellen, wird der Fehler im Rahmen der Rechengenauigkeit zu Null. Dies war zu erwarten, da für diese Punkte die Einflußgleichungen zur Bestimmung der unbekannten Spannungen aufgestellt wurden. Das heißt für diese Punkte erhält man genau die Gleichungen, die schon als Zeilen in das Gleichungssystem eingingen, das im Rahmen der Kollokation aufgestellt wurde (vgl. 3.4.1: Aufstellen des Gleichungssystems).
**Zu Abbildung 20 bis Abbildung 23:**

Beim Dreiecksquerschnitt wird bereits mit 15 Elementen eine sehr hohe Genauigkeit erreicht. Gründe hierfür sind, daß keine einspringenden Ecken vorhanden sind und daß es sich um einen gedrungenen Querschnitt handelt.


Es läßt sich auch gut ablesen, warum in den Fehlerverläufen im doppelt-logarithmischen Maßstab die Konvergenzkurve flacher verläuft als beispielsweise beim Dreieck (vgl. *Abbildung 19*): Die lineare Randeinteilungsverfeinerung zeigt hier deutlich weniger Erfolg.
Abbildung 20: Dreieck, 3 Elemente, Residuum r

Abbildung 21: Dreieck, 15 Elemente, Residuum r
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 22: I-Profil, 12 Elemente, Residuum r

Abbildung 23: I-Profil, 52 Elemente, Residuum r
6 Adaptive Randelement-Teilung

Unter „Adaption“ versteht man die Verfeinerung eines Netzes unter Berücksichtigung der Tatsache, daß der Fehler, mit dem numerische Verfahren behaftet sind, in der Regel ortsabhängig ist. Ziel der Adaption ist es, dort eine Netzverfeinerung vorzunehmen, wo der Fehler verhältnismäßig groß ist, um so Schritt für Schritt bessere Ergebnisse zu erhalten, ohne unnötig feine Netzeinteilungen in unkritischen Gebieten „mitzuziehen“.

Da bei der BE-Methode nur der Rand, d.h. nur eine Dimension in unterschiedlich große Elemente eingeteilt werden muß, bietet sich ein adaptives Vorgehen zur Verbesserung der Ergebnisse an. (Bei finiten Elementen ist im Falle des zweidimensionalen Membranproblems die unregelmäßige, problemorientierte Vernetzung ungleich schwieriger zu realisieren.)

Das Ergebnis für das Torsionswiderstandsmoment soll also schrittweise verbessert werden, indem der Rand immer feiner aufgeteilt wird. Hierbei werden nur die Elemente weiter zerlegt, die im Vergleich zu den anderen Elementen große Fehler ins Endergebnis einfließen lassen. Um eine adaptive Methode implementieren zu können, muß also ein Kriterium für die Größe des Fehlers eines jeden Elementes gefunden werden. Hier bietet sich das Residuum der Verschiebung an, wie nachfolgend gezeigt wird.

6.1 Residuum der Verschiebung als Kriterium für Adaption

Betrachtet man den Verlauf des Residuums in Abbildung 23, fällt auf, daß der Fehler keineswegs für jedes Element gleich groß ist. So wird im Bereich der Stegmitte (bei obiger Einteilung in 72 Randelemente) das Residuum so klein, daß hier eine feinere Einteilung des Randes überflüssig wird. Dagegen sind die Eckelemente mit großem Fehler behaftet. Um einen möglichst guten Wert für das Torsionsträgheitsmoment zu erhalten, führt man die Berechnung nun mehrmals für jeweils feinere Randerteilung aus. Geht man hierbei wie bisher vor, d.h. der Rand wird streng nach dem linearen Schema (d.h. alle Elemente seien ungefähr gleich lang) diskretisiert, wird der Rand auch dort feiner aufgeteilt, wo das Residuum vergleichsweise klein ist. Das bedeutet, daß der Rechenaufwand durch diese zusätzlichen Elemente erhöht wird, ohne daß sich das Ergebnis entscheidend verbessert. Es scheint also sinnvoll, den Rand nur dort feiner aufzuteilen, wo dies eine merkliche Verbesserung des Ergebnisses verspricht.
Hat man in einem ersten Rechendurchgang den Verlauf des Residuums bestimmt, kann daraus ein Kriterium für die Randinteilung im nächsten Rechendurchgang gewonnen werden. Dazu wird ein absoluter Wert für jedes Element bestimmt, und zwar das Integral des Residuums $I_R$ über das jeweilige Element:

$$I_R = \int_I r_{st}(y) \, d\Gamma_y$$

Aus allen $I_{Ri}$ wird der maximale Wert $I_{R,max}$ bestimmt, der als Kriterium für eine Elementteilung dient: Überschreitet der Wert $I_{Ri}$ einen festgelegten Prozentsatz von $I_{R,max}$ (z.B. 50% $I_{R,max}$), wird dieses Element für den nächsten Rechendurchgang geteilt.


### 6.2 Programmierung der Adaption

Um die Adaption zu realisieren, ist natürlich eine „äußere Schleife“ erforderlich, die die Berechnung mehrmals durchführt. In der Schleife sind dann stets zwei Hauptaufgaben auszuführen:

- Berechnung des Problems unter aktueller Randinteilung mittels `ComputeTorque`
- Bestimmen der neuen Randinteilung durch `DivideForUError`, also Teilen der Elemente, sofern sie den maximalen zulässigen Wert für das Residuum übersteigen.

Als Abbruchbedingung kann z.B. eine maximal zulässige Elementanzahl angegeben werden. Sinnvoll wäre auch, eine Unterschreitung eines Fehlerwertes (siehe Abschnitt 5: Fehlergrößen) als Abbruchbedingung zu verwenden.
Das Hauptprogramm für die Adaption sieht so aus:

```cpp
void main()
{
  ifstream in("tprof.txt");
  CBModel* pBM=new CBModel();
  pBM->Load(in);
  CColloc C(pBM);
  double a;
  while (pBM->GetSize()<200)
  {
    a=C.ComputeTorque();
    cout << "Torque:" << a << endl;
    C.DivideForUError(.5,MAX);
  }
}
```

6.3 **Beurteilung des adaptiven Vorgehens**

Das adaptive Vorgehen kann wie folgt beurteilt werden:

- Um dieselbe Ergebnisgenauigkeit wie bei linearer Einteilung des Randes zu erhalten, sind im Vergleich deutlich weniger Elemente erforderlich. Dies hat zur Folge, daß die Konvergenzkurve im doppelt-logarithmischen Maßstab steiler verläuft. Siehe hierzu *Abbildung 30* und *Abbildung 32*.


- Je nach Beschaffenheit des Querschnittes ist die Anwendung der Adaption mehr oder weniger sinnvoll. Die Untersuchung zeigt jedoch, daß gerade bei komplizierteren Querschnittsformen wie dem I-Profil, sowie später bei gelochten Querschnitten, eine Adaption weit schneller zu genauen Ergebnissen führt.
Abbildung 24: Adaptionsschritt, 8 Elemente

Abbildung 27: Adaptionsschritt, 22 Elemente

Abbildung 25: Adaptionsschritt, 11 Elemente

Abbildung 28: Adaptionsschritt, 32 Elemente

Abbildung 26: Adaptionsschritt, 15 Elemente

Abbildung 29: Adaptionsschritt, 44 Elemente
Abbildung 30: Vergleich adaptive/lineare Elementteilung Dreieck und Rechteck

Der Fehler gegen den Referenzwert konvergiert schneller, d.h. weniger Elemente sind bei adaptiver Teilung notwendig, um die gleiche Ergebnisgenauigkeit für das Torsionswiderstandsmoment zu erhalten.
Hier wird der Unterschied noch deutlicher:
Besonders beim I-Profil, wo die Konvergenzkurve aufgrund der stark unterschiedlichen Elementgrößen zunächst flach verläuft, ist bei adaptivem Vorgehen ein früheres Übergehen in die endgültige Steigung festzustellen. Außerdem verlaufen die Kurven für die Adaption deutlich steiler.
Zu den Abbildungen:
Das adaptive Verfahren liefert bei komplexeren Querschnitten mit einspringenden Ecken deutlich bessere Ergebnisse als eine lineare Elementteilung. Beim I-Profil waren in absehbarer Zeit kaum vernünftige Lösungsergebnisse zu erhalten; dies war aufgrund der langsamen Konvergenz (siehe Abbildung 18) auch zu erwarten.

Fazit: Das adaptive Verfahren, basierend auf dem Residuum der Randverschiebung, ist auf jeden Fall der linearen Randverfeinerung vorzuziehen.
7 Das Galerkin-Verfahren

Bislang wurden alle BE-Berechnungen mit Hilfe des Kollokationsverfahrens durchgeführt. Dabei wurde ein Gleichungssystem zur Bestimmung der unbekannten Randgrößen-Koeffizienten aufgestellt, indem die Einflußgleichung für die einzelnen Kollokationspunkte exakt erfüllt werden sollte.

Einen anderen Weg geht das Galerkin-Verfahren:
„Beim Galerkin-Verfahren werden die Koeffizienten so bestimmt, daß der Defekt im L₂-Sinne orthogonal zu den Ansatzfunktionen ist.“ [1], S.50.

Dazu wird im Falle der vollständig eingespannten Membran das Residuum der Verschiebung, gewichtet mit den Ansatzfunktionen, über den gesamten Rand integriert. Dieses Integral muß dann entsprechend der Vorgabe Orthogonalität zu Null werden.


Der große Nachteil des Galerkin-Verfahrens besteht darin, daß der Rechenaufwand erheblich größer wird zum Erstellen der Matrizen im Gleichungssystem (zum Bestimmen der unbekannten Randgrößen) durch die Integration über den Rand. Ob die Vorteile des Galerkin-Verfahrens diesen erhöhten Aufwand rechtfertigen, wird im letzten Abschnitt dieses Kapitels untersucht.
7.1 Realisierung des Galerkin-Verfahrens

Der Defekt im $L_2$-Sinne soll orthogonal zu den Ansatzfunktionen werden.
Orthogonalität wird im Zweidimensionalen rechnerisch nachgewiesen, indem das Vektorprodukt Null ergeben muß.

\[ \int_{\Gamma} r_u(z) \varphi(z) d\Gamma_z = 0 \]

Den Defekt stellt, sofern man mit der 1. Einflußgleichung für die Membran Gleichung 3 arbeitet, das Residuum der Verschiebung dar. Für eine konstante vorgegebene Verschiebung $u(z) = 0$ gibt Gleichung 3 direkt das Residuum an (vgl. auch 5.2: Fehler bezüglich der Verschiebung: $u^2$ - Residuum).

Berücksichtigt man nun noch, daß die Ansatzfunktion der Spannung elementweise konstant ist, gilt:

Gleichung 22
\[ \int_{\Gamma} u(z) d\Gamma_z = 0 \]

Das Integral in Gleichung 22 berechnet man mittels Gauss-Legendre-Quadratur, indem für Einzelpunkte die Verschiebung $u$ mit Gleichung 10 bestimmt, und dann mit der entsprechenden Wichtung nach Gauss-Legendre multipliziert.

Gleichung 10
\[ \frac{1}{2} u(x) = \int_{\Gamma} \left[ \left( - \frac{1}{2\pi N} \ln r \right) \varphi(y) + \frac{1}{2\pi} \frac{r_n}{r} \varphi(y) \right] ds_y . \]

Im Gegensatz zur Kollokation liegt dieser Punkt nun nicht mehr stets in Elementmitte (vergleiche Abschnitt 3.4.2: Lösen der Integrale), sondern entsprechend der Legendre-Punkte über das Element verteilt. Für die analytische Integration muß dies natürlich entsprechende Erweiterungen zur Folge haben, die diese neue Konstellation berücksichtigen.

Da der Träger der Ansatzfunktionen jeweils auf nur ein Element beschränkt ist, erhält man ein Gleichungssystem, das sich nur geringfügig von dem der Kollokation unterscheidet. Es muß also nicht über den gesamten Rand integriert werden, sondern nur jeweils auf dem Element, für das die Ansatzfunktion aktiv ist. Jede Zeile repräsentiert nun nicht einen Kollokationspunkt, sondern ein „aktives“ Element.
Aufgrund der Verwandtschaft der Verfahren Kollokation und Galerkin konnte ich das Galerkin-Verfahren durch eine von der Klasse CColloc abgeleiteten Klasse implementieren, die den Namen CGalerkin erhielt. Hierbei war es nur nötig, zwei Funktionen zu überladen:

Die Berechnung der Matrizen und Vektoren innerhalb des Gleichungssystems änderte sich, was eine Neundefinition der Funktion ComputeHGP erforderlich machte. Nachfolgend habe ich die beiden virtuellen Funktionen ComputeHGP der Klassen CColloc und CGalerkin mit Erläuterungen gegenübergestellt.

Außerdem mußte die Funktion Compute überladen werden, da die Lösung des Gleichungssystems, das zwar immer noch die Form $\mathbf{H}\hat{\mathbf{u}} = \mathbf{G}\hat{\mathbf{p}}$ hat, nun mit der Funktion SolveCholesky ausgeführt werden konnte. Das Cholesky-Verfahren nützt aus, daß die Matrix $\mathbf{G}$ symmetrisch ist.

```cpp
void CColloc::ComputeHGP(int ng)
{
    // Variablendeklaration:
    CElement CE;
    double h1,h2,g,pn;
    int j,i;
    // Matrix H initialisieren:
    H->Clear();
    // Vektor p initialisieren:
    for (i=0;i<Size;i++) p[i]=0;
    // Zeiger auf Elemente initialisieren:
    CBEdge* px=pBM->GetLastEdge();
    CBEdge* py=pBM->GetLastEdge();
    // Schleife über alle Elemente
    i=1;
    while (i<=Size)
    {
        j=1;
        while (j<=Size)
        {
            // Integrationselement initialisieren:
            CE.Init(this,py,i);
            // Länge des Elements berechnen
            le=0.5*px->GetLength()*Factor;
            // Integralwerte berechnen, für Kollokationspunkt auf px in Elementmitte (0.5):
            CE.ComputeIntegralsHGP(ng,px,0.5,h1,h2,g,pn);
            // errechnete Werte eintragen:
            (*H)(j,i)+=h2;
            (*H)(j,(i%Size)+1)+=h1;
            (*G)(j,i)=g;
            p[j-1]+=pn;
            px=px->GetNextEdge();
            j++;
        }
        py=py->GetNextEdge();
        i++;
    }
    // 0.5 in Diagonale addieren
    for (i=1;i<Size;i++) (*H)(i,i)+=0.5;
}
void CGalerkin::ComputeHGP(int ng)
{
    CElement CE;
    double h1,h2,g,pn;
    double t,le,r;
    int j,i,k;
    H->Clear(); G->Clear();
    for (i=0;i<Size;i++) p[i]=0;
    CBEdge* px=pBM->GetLastEdge();
    CBEdge* py=pBM->GetLastEdge();
    // Schleife über alle Elemente
    i=1;
    while (i<=Size)
    {
        j=1;
        while (j<=Size)
        {
            CE.Init(this,py,i);
            // Transformation von Intervall [-1,1] auf [0,1]
            t=(t+1.)*0.5;
            // Integralwerte berechnen, für Aufpunkt auf Element px, t zwischen 0 und 1:
            CE.ComputeIntegralsHGP(ng,px,t,h1,h2,g,pn);
            // Werte mit Gauss-Wichtung und Elementlänge multiplizieren:
            r=w*le;
            (*H)(j,i)+=r*h2;
            (*H)(j,(i%Size)+1)+=r*h1;
            (*G)(j,i)=r*g;
            p[j-1]+=r*pn;
            px=px->GetNextEdge();
            j++;
        }
        py=py->GetNextEdge();
        i++;
    }
    // 0.5 in Diagonale addieren
    for (i=1;i<Size;i++) (*H)(i,i)+=0.5;
}
```
Praktisch ist, daß beide Funktionen *ComputeHGP* auf dieselbe Klasse *CElement* zurückgreifen können. Die Memberfunktion *ComputeIntegralsHGP* von *CElement* enthält zwar Funktionalitäten, die bei der Aufstellung des Gleichungssystems in der Kollokation noch nicht benötigt werden. Diese sind die analytische Berechnung der Integrale für eine beliebige Lage des Aufpunktes auf dem Integrationselement. Störend ist dies jedoch nicht, da dies lediglich eine zusätzliche, rechenzeitlich völlig unbedeutend wirkende if-Abfrage mit sich bringt. Für die Berechnung der Verschiebungsgrößen an beliebigen Punkten, wie sie bei Bestimmung des Residuums benötigt wird, ist die Möglichkeit einer beliebigen Lage des Aufpunktes auf dem Integrationselement sowieso auch bei der Kollokation notwendig.

Die entsprechenden Programmteile der Memberfunktion *ComputeIntegralsHGP* für den Fall, daß der Aufpunkt auf dem Integrationselement liegt, sind nachfolgend dargestellt:

```
// Wenn Aufpunkt-Element gleich Integrationselement, dann:
if (pxE==pE)
{
// Matrizenwerte für H und P sind Null:
    H1=0;
    H2=0;
    P=0;
// Liegt Aufpunkt in Elementmitte ? (Für Kollokation; schnell rechen und Funktion verlassen!)
    if (fabs(dx-0.5)<EPS)
    {
// Länge des Integrationselementes holen:
      le=GetScLength();
// Integralwert für Matrix G berechnen, vgl. Gleichung 14
      G=-le*(log(le)-log(2)-1)/(2*PI*pC->GetNormalForce());
      return;
    }
else
// Aufpunkt liegt nicht in Elementmitte:
{
    G=0;
// Integral berechnen; erst von 0 bis dx, dann von dx bis 1 integrieren:
    le=GetScLength()*dx;
    if (fabs(le)>0)
        G=-le*(log(le)-1);
    le=GetScLength()*(1-dx);
    if (fabs(le)>0)
        G+=-le*(log(le)-1);
    G/=(2*PI*pC->GetNormalForce());
    return;
}
```
7.2 Bewertung


Auf der nachfolgenden Seite ist dann der relative Fehler des Torsionsträgheitsmoments für das Galerkin-Verfahren im Vergleich zur Kollokation dargestellt (Abbildung 39). Man erkennt, daß sich im doppelt-logarithmischen Maßstab wiederum eine geradenförmige Konvergenz einstellt. Diese Gerade ist jedoch nicht steiler, sondern lediglich leicht nach unten verschoben, d.h. die Ergebnisse sind zwar immer leicht besser, verbessern sich aber nur in derselben Größenordnung mit wachsender Elementzahl.

Abbildung 40 und Abbildung 41 vergleichen Kollokation und Galerkin-Verfahren in Bezug auf die Rechenzeit. Die benötigte Rechenzeit wird maßgebend durch zwei Arbeitsschritte beeinflußt:

- das Lösen des so aufgestellten Gleichungssystems.

Abbildung 40 zeigt die Zusammenhänge für Kollokation mit einem Scaled-Row-Gauss-Gleichungslöser auf. Hier wird deutlich: je weniger Elemente in die Berechnung eingehen, desto weniger fällt auch die Lösung des Gleichungssystems ins Gewicht. Die Kurve der benötigten Gesamtzeit verläuft also für eine geringe Anzahl Elemente nahe der (im doppelt-logarithmischen Maßstab als Gerade erscheinenden) Kurve der benötigten Zeit für das Berechnen der Matrizienwerte. Bei ca. 700 bis 800 Elementen schneiden sich die beiden Trendlinien für Gleichungslösung und Matrizienwerte-Berechnung. Hier tragen diese Arbeitsschritte
Arbeitsschritte denselben Anteil an der Gesamtzeit. Für noch feinere Randeinteilungen schmiegt sich der Verlauf der Gesamtzeit dann immer mehr an die Trendlinie der Gleichungslösung an.

_Abbildung 41_ zeigt das Galerkin-Verfahren, das für die Lösung des erzeugten Gleichungssystems auf das Cholesky-Verfahren zurückgreift, da die Matrix \( G \) hier symmetrisch ist.

Im Vergleich zur vorigen _Abbildung 40_ ist nun die Trendlinie „Matrizenwerte berechnen“ im doppelt-logarithmischen Maßstab nach oben verschoben. Wie weit sich diese Trendlinie nach oben verschiebt, hängt von der Anzahl der Gauss-Legendre-Punkte ab, über die das Integral des Residuums bestimmt wird. Das Schaubild zeigt hier die Kurve für 8 GL-Punkte. Bei \( n \) Gausspunkten muß die \( n \)-fache Menge an Integralen gegenüber der Kollokation bestimmt werden. Die Berechnung der Matrizenwerte muß also bei 8 GL-Punkten ungefähr 8 mal so lange wie bei der Kollokation dauern.

Die Trendlinie „LGS lösen“, wobei nun das Cholesky-Verfahren angewendet wurde, verläuft im Schaubild in gleicher Steile wie beim Scaled-Row-Gauss-Verfahren, da sich die Konvergenzordnung nicht ändert, der Aufwand wächst nach wie vor mit \( n^2 \) an.

Der Zeitgewinn von Cholesky gegenüber dem die volle Matrix verwendenden Verfahren fiel im Testlauf, wie das Schaubild zeigt, sogar deutlich geringer als der erwartete Faktor 2 aus.

Durch die länger dauernde Berechnung der Matrizenwerte und die kürzere Zeit für die Gleichungslösung verschiebt sich der Schnittpunkt der beiden Trendlinien nach rechts, bei 8 GL-Punkten bis ca. 3500 Elementen.
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 35: Dreieck mit Galerkin, 3 Elemente, Residuum r

Abbildung 36: Dreieck mit Galerkin, 3 Elemente, Residuum r
Abbildung 37: I-Profil mit Galerkin, 12 Elemente, Residuum r

Abbildung 38: I-Profil mit Galerkin, 52 Elemente, Residuum r
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 39: Fehlervergleich Galerkin / Kollokation; lineare Elementteilung
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 40: Rechenzeit Kollokation mit Scaled-Row-Gauss-Gleichungslöser

Abbildung 41: Galerkin und Cholesky-Gleichungslöser
Fazit:

Die Lösung des Gleichungssystems wird erst jenseits von 700-800 Elementen die vorherrschende Zeitgröße, beim Galerkin-Verfahren je nach Anzahl der Gauss-Legendre-Punkte entsprechend später. Im Falle von 8 Gausspunkten kann man den Schaubildern entnehmen, daß das Galerkin-Verfahren erst jenseits der 4000 Elemente überhaupt schneller als die Kollokation werden kann, egal, wie schnell der Gleichungslöser arbeitet. Bei 4 Gausspunkten, was aus Genauigkeitsgründen als Minimum für die Bestimmung der Integrale angesehen werden kann, kann man diese Grenze frühestens bei 2000 Elementen erwarten.

Diese Erkenntnisse gelten übrigens unabhängig von der Rechenleistung, da auf schnelleren Rechnern beide bestimmenden Trendlinien im doppelt-logarithmischen Maßstab dieselbe Verschiebung nach unten erfahren würden (bei einer Steigerung der Rechenleistung um den Faktor 10 z.B. um 1 Segment im Schaubild.)

Da außerdem die Qualität der Ergebnisse laut Abbildung 39 mit Galerkin nur geringfügig besser als die Kollokation ist, scheint das Galerkin-Verfahren nur bei Randerteilungen sinnvoll, die Elementzahlen im über dreistelligen Bereich verarbeiten. Da für ingenieurmeßige Zwecke bereits vorher eine ausreichende Genauigkeit erreicht wird, ist das Kollokationsverfahren auf jeden Fall die bessere Wahl. Dies gilt insbesondere, wenn adaptive Lösungsansätze gewählt werden, da hier die benötigte Elementzahl deutlich verringert wird (vergleiche auch Abbildung 31).

Ein weiterer Punkt, der aus diesem Vergleich hervorgeht : das Cholesky-Verfahren schneidet im Vergleich zum Scaled-Row-Gauss-Verfahren kaum besser ab. Möchte man die symmetrischen Matrizen-Eigenschaften konsequent nutzen, sollte auf jeden Fall eine Verwendung iterativer Gleichungslöser in Betracht gezogen werden.

Anmerkung: Die beiden verwendeten Gleichungslöser wurden mir von Prof. Holzer zur Verfügung gestellt, und unverändert übernommen.
8 Querschnitte mit Löchern

Keine Berücksichtigung erfuhren bis jetzt Querschnitte mit Löchern. Da jedoch Hohlquerschnitte sehr gebräuchlich sind, und eine Ermittlung der Torsionsträgheitsmomente nur für bestimmte dünnwandige Querschnitte per Formeln möglich ist, wäre natürlich eine entsprechende Erweiterung wünschenswert.

Dieses Vorhaben gestaltet sich jedoch relativ aufwendig:
Im Gegensatz zur Bestimmung der Flächenträgheitsmomente 2. Ordnung kann hier der Querschnitt nicht mit einem Polygonzug umschrieben werden. Der Querschnitt kann nämlich nicht einfach „aufgeschnitten“, und das Loch im gegenläufigen Sinne (zum äußeren Rand) durchlaufen werden, um dessen Geometrie zu erfassen. Statt dessen müssen mehrere Ringlisten sowohl den äußeren Rand als auch die inneren Ränder, sprich die Ränder der Löcher, eindeutig festlegen. Die Berücksichtigung mehrerer Ringlisten wird also für die Berechnung der Torsionsträgheitsmomente notwendig. Auf diese Problematik wird in Abschnitt 8.1 eingegangen.

Das ist jedoch nicht die einzige Änderung, die vorgenommen werden muß: die Wahl der Randverschiebung der Löcher ist nicht mehr beliebig, kann also nicht mehr frei zu Null gewählt werden, damit das nach der Membrananalogie berechnete 2-fache Volumen unter der Membran noch dem Torsionswiderstandsmoment entspricht. Hintergründe und Lösungsansatz hierzu werden in Abschnitt 8.2 behandelt.

Der erste Querschnitt weist ein Loch auf, das ausmittig gewählt wurde.
8.1 Mehrere Ränder

Um die Löcher erfassen und verarbeiten zu können, reicht die bislang verwendete Struktur der Ringliste nicht mehr aus. Es muß eine übergeordnete Datenstruktur eingeführt werden, die mehrere Ringlisten beinhaltet. Eine Ringliste stellt dann den äußeren Rand dar, die anderen Ringlisten begrenzen die Löcher, sind also innere Ränder.
8.1.1 Liste zur Verwaltung der Ränder


Abbildung 45 zeigt das neue Datenmodell für den Querschnitt des 6-Ecks mit 2 Löchern in anschaulicher Form. Die Ringlisten selbst weisen dieselbe Gestalt auf, wie dies für ungelochte Querschnitte bereits dargestellt wurde.

Abbildung 45: Datenmodell-Beispiel: gelochte Querschnitte

8.1.2 Kollokation mit mehreren Ringlisten

Eine neue Klasse für die Kollokation, die beliebig viele Ringlisten verarbeiten kann, stellt die Klasse \( CExColloc \) dar (für extended collocation). Dem Konstruktor wird nun nicht mehr ein
Zeiger auf eine Ringliste übergeben, sondern ein Zeiger auf eine Liste von Zeigern auf die Ringlisten mit den Randdaten.

Auch die Klasse CExColloc wurde, wie schon CColloc, von der Klasse CGeneralColloc abgeleitet. Dies gewährleistet, das die Klasse CElement unverändert weiterverwendet werden kann; manche Memberfunktionen von CElement, wie z.B. die Bestimmung der Verschiebung $u(x)$, müssen nämlich auf Funktionen der Kollokationsklasse zurückgreifen.

Hier zur Veranschaulichung ein kurzer Auszug aus der Headerdatei von CExColloc:

```cpp
// CGeneralColloc wird benötigt:
#include "gencoll.h"
// Listenklasse für die Verwaltung mehrerer Ringlisten
#include "list.h"

class CExColloc : public CGeneralColloc
{
    ...
    protected:
    // Liste von Zeigern auf CBModel-Objekte (=Ringlisten)
    List<CBModel>* pML;
    ...
    public:
    // Konstruktor:
    CExColloc(List<CBModel>* pML);
    // die nach wie vor zentralen Funktionen:
    virtual void ComputeHGP(int ng);
    void Compute();
};
```

Die Handhabung von CExColloc unterscheidet sich, abgesehen vom neuen Konstruktor, nicht von CColloc. Für die Memberfunktionen wurden durchgängig dieselben Bezeichnungen gewählt. Möchte man im Programm also zunächst auf Querschnitte mit Löchern verzichten, ist eine spätere Umstellung auf die Klasse mit entsprechend erweiterter Funktionalität ohne großen Aufwand möglich.

**Was ändert sich bei den Berechnungen infolge dieser beliebig vielen Ringlisten, die das Gebiet der Membran begrenzen?**

**Punkt 1**

Die Berechnung der unbekannten Randdaten steuert die Funktion Compute. Hier ändert sich im Vergleich zur Klasse CColloc nur wenig (vergleiche Abschnitt 3.5). Die Matrizen und
Vektoren müssen natürlich entsprechend der gesamten Elementzahl dimensioniert werden. Die aufgebrachte Randverschiebung wird zunächst konstant bei Null belassen und in der Unterfunktion ComputeVectorsUT, wie bei CColloc, gesetzt. Dann folgt, wie gehabt, die Aufstellung der Matrizen und Vektoren (dazu mehr unter Punkt 2), sowie das Lösen des linearen Gleichungssystems.

Punkt 2

Betrachten wir nun die Funktion ComputeHGP, in der das Gleichungssystem für die Bestimmung der unbekannten Randdaten aufgestellt wird. Auf jeden Kollokationspunkt hat jedes Element Einfluß, egal, ob das Element auf demselben Teilrand liegt oder nicht. Da außerdem auf jedem Element aller Teilränder ein Kollokationspunkt zu liegen kommt (und zwar in Elementmitte beim gewählten Ansatz für stückweise konstante Spannungen), wird daher eine doppelt geschachtelte Schleife benötigt, die über sämtliche Teilränder läuft.


Beachtet werden muß auch, daß alle Teilrand-Daten korrekt in die Matrizen eingetragen werden: Kritisch ist hier immer der Übergang von den ringförmigen Teil-Strukturen in die nicht geschlossenen Matrizen-Teile, da ja eine Ringliste keinen Anfang bzw. kein Ende hat, die Spalten der Matrix hingegen schon; hier muß bei den Zählern für die laufende Spalte der Matrix entsprechend achtgegeben werden. Hierzu sei nochmals die Gleichung für die Matrixelemente von H bei Querschnitten ohne Löcher angeführt, die ansonsten unverändert für gebohnte Querschnitte übernommen werden kann:

\[
H_0 = \frac{1}{2} \delta_i - \frac{1}{2\pi} \left[ \int_{\xi_2}^{r_n} (1 - \frac{\tilde{\xi}_1}{l_j}) d\xi_1 + \int_{\xi_{j-1}}^{r_n} \frac{\tilde{\xi}_2}{l_j} d\tilde{\xi}_2 \right]
\]

i ist hierbei nach wie vor die Variable, die die Zeile der Matrix bestimmt; jede Zeile repräsentiert einen Kollokationspunkt. j läuft über die Integrationselemente; für j − 1 = 0 muß nun der
nun der Zähler auf das „letzte“ (bezogen auf bei beliebigem Element begonnene, dann sequentielle Numerierung) Randelement des Teil-Randrings gesetzt werden.

Zu diesen Ausführungen ein Auszug aus der entsprechend erweiterten Funktion *ComputeIntegralsHGP* von *CExColloc*:

```cpp
void CExColloc::ComputeHGP(int ng)
{
    // Variablendefinitionen etc.
    ...
    // Iteratoren für Liste der Teilränder (Ringlisten):
    Position pos1;
    Position pos2;
    // zwei Zeiger auf Teilränder:
    CBModel* pM1;
    CBModel* pM2;
    // Iterator initialisieren:
    pos1=pML->GetHeadPosition();
    k=0;
    // äußere Schleife über alle Teilränder:
    while (pos1)
    {
        // Zeiger auf neuen Teilrand holen:
        pM1=pML->GetNext(pos1);
        // Größe des Teilrandes holen:
        Size1=pM1->GetSize();
        l=0;
        pos2=pML->GetHeadPosition();
        // innere Schleife über alle Teilränder:
        while (pos2)
        {
            pM2=pML->GetNext(pos2);
            Size2=pM2->GetSize();
            i=1;
            // Zeiger auf Elemente initialisieren:
            px=pM1->GetLastEdge();
            py=pM2->GetLastEdge();
            // äußere Schleife über alle Elemente für Integrationselemente:
            while (i<=Size2)
            {
                j=1;
                // innere Schleife über alle Elemente für Kollokationspunkte:
                while (j<=Size1)
                {
                    CE.Init(this,py,i+l);
                    CE.ComputeIntegralsHGP(ng,px,0.5,h1,h2,g,pn);
                    (*H)(j+k,i+l)=h2;
                    (*H)(j+k,(i%Size2)+1+l)=h1;
                    (*G)(j+k,i+l)=g;
                    p[j+k-1]+=pn;
                    CE.Init(this,px,i+1);
                }
            }
        }
    }
    // Hier modulo-Befehl, um an richtige Stelle einzutragen (siehe Text):
    (*H)(j+k,(i%Size2)+1+l)=h1;
    (*G)(j+k,i+l)=g;
    p[j+k-1]+=pn;
    // nächstes Element holen:
    px=px->GetNextEdge();
    j++;
    // nächstes Element holen:
}
```
Schlußfolgerungen:

Mit diesen Änderungen, sowie einer Anpassung aller übrigen (Hilfs-)Funktionen in ähnlicher Art, d.h. die Verarbeitung mehrerer Ringlisten durch eine äußere Schleife, können nun Membranen mit beliebig vielen Löchern berechnet werden. Die Wahl der aufgebrachten Randverschiebung ist hierbei beliebig, speziell vom Programm unterstützt werden beliebige Verschiebungsfunktionen, die mit `SetFunction` dem Kollokationsmodell zugeordnet werden können. Als Belastung wurde im Hinblick auf die Problemstellung (Bestimmung des Torsionswiderstands moments) nur gleichmäßiger, konstanter Druck vorgesehen. Per Voreinstellung ist dieser Druck \( p = 2 \), mit `SetPressure` können andere Drücke gesetzt werden.

Anmerkung:

Der Spezialfall, daß die Verschiebung des gesamten Randes konstant zu Null gewählt wird, kann nicht mehr zur Bestimmung des Torsionswiderstands moments verwendet werden. Das zweifache Volumen unter dieser Membran liefert nämlich nicht das gesuchte Ergebnis. Trotzdem kann eine Berechnung der Membran unter diesen Randbedingungen zur Überprüfung der bisher vorgenommenen Erweiterungen (gegenüber der Implementierung für ungelochte Querschnitte) dienen.

8.1.3 Überprüfung der Berechnungen

Prinzipiell kann wie unter Abschnitt 3.6 vorgegangen werden, um die berechneten Spannungswerte zu verifizieren. Dies wurde im Rahmen der Arbeit auch vorgenommen, ich möchte an dieser Stelle jedoch auf nochmals ausführliche Auswertung der Ergebnisse verzichten.

Darstellen möchte ich kurz den Verlauf des Fehlers gegen Referenzwert für das 2-fache Volumen unter der Membran mit konstanter Randverschiebung gleich Null, da sich hier noch-
nochmals die Möglichkeit bietet, mit per FE-Methode berechneten Referenzwerten zu vergleichen. Siehe hierzu Abbildung 46, sowie Tabelle 13 bis Tabelle 15.

Abbildung 46: Fehler gegen Referenzwert für Lochquerschnitte mit Randverschiebung const.=0

8.2 Vom Membranvolumen zum Torsionswiderstandsmoment

Wie in der Einleitung zu Abschnitt 8 bereits angedeutet, ist mit der Lösung des Problems der gelochten Membranen noch nicht die gesamte Arbeit getan. Um wieder vom Volumen unter der mit dem konstanten Druck \( p = 2 \) belasteten Membran auf das Torsionsträgheitsmoment schließen zu können, müssen für die Innenränder bestimmte Bedingungen eingehalten werden, und zwar:

Die Bredtsche Formel muß für alle Innenränder erfüllt werden (vgl. [1], S.129):

\[
\int_{\Gamma} \frac{\partial u}{\partial n} ds = -2A,
\]

\[
I_i = \int_{\Gamma_i} \frac{\partial u}{\partial n} ds = -2A_i
\]
In Worten ausgedrückt: Mit Gleichung 2 und \( N = 1 \) steht im Integral die Spannung \( t \). Da \( t \) über den jeweiligen Innenrand integriert wird, und auf der rechten Seite die Fläche \( A_i \) steht, die vom Rand \( \Gamma_i \) begrenzt wird, mit dem aufgebrachten Druck \( p = 2 \) multipliziert wird, kann Gleichung 23 als Kräftegleichgewichtsbedingung aufgefaßt werden.

Weiterhin gilt: Die Verschiebung der Ränder sei nach wie vor konstant.

Um Gleichung 23 zu erfüllen, ohne die übrigen Gleichungen der Membrantheorie zu verletzen, kann nun wie folgt vorgegangen werden:

\[
\text{Abbildung 47: Beispiel für Horizontalverschiebung der Löcher}
\]

Die Randverschiebungen für die Löcher muß so eingestellt werden, daß das Integral über die Randspannungen entlang eines jeden Loches genau der negativen zweifachen Fläche des Loches entspricht. Abbildung 47 zeigt eine angenommene Konstellation für konstante Verschiebungen der Löcher und damit konstanter horizontaler Auslenkung des Randes.

Das Problem hierbei ist jedoch, daß diese horizontalen Auslenkungen zunächst nicht bekannt sind. Für \( n \) Löcher sind also \( n \) Konstanten zu bestimmen.

\[\text{8.2.1 Methode nach Griffith und Taylor}\]

Um diese Schwierigkeit zu bewältigen, geht man nach der Methode von Griffith und Taylor vor, die in [2], S.331 kurz, aber sehr präzise beschrieben wird.

Die Prantlsche Spannungsfunktion \( \phi \) wird, so die Idee von Griffith und Taylor, aus mehreren Teilfunktionen \( \phi_i \), entsprechend der Anzahl \( n \) der Ränder, zusammengesetzt.

In der Membrananalogie stellt die Prantlsche Spannungsfunktion die Verschiebungsfunktion \( u \) dar, die Teilfunktionen werden im folgenden mit \( u_i \) bezeichnet.
Die Funktion $\phi (= u)$ wird folgendermaßen gebildet:

\begin{equation}
\phi = m_1 \phi_1 + m_2 \phi_2 + \cdots + m_n \phi_n
\end{equation}

$m_i$ sind numerische Faktoren mit der Bedingung:

\begin{equation}
m_1 + m_2 + \cdots + m_n = 1
\end{equation}

Damit wird erreicht, daß die verschiedenen, zunächst unbekannten Horizontalverschiebungen der Löcher der Gesamtfunction $\phi$ bzw. $u$ leichter eingestellt werden können, so daß sie die Randbedingung \textit{Gleichung 23} nicht verletzen.

Die Teilfunktionen unterscheiden sich untereinander nur in sofern, daß für die Verschiebungen der inneren Ränder verschiedene günstig gewählte Konstanten angesetzt werden.

Günstig gewählt heißt hier: jeweils ein Loch $L_i$ erhält eine konstante Verschiebung \textit{ungleich Null}, alle übrigen Löcher erhalten eine konstante Verschiebung \textit{gleich Null} (Nicht zwingend notwendig, aber sinnvoll. Entscheidend ist im Prinzip nur, daß jeweils verschiedene Kombinationen gewählt werden, um nachher ein linear unabhängiges Gleichungssystem zu erhalten).

Man führt nun für alle Teilfunktionen unter diesen Randverschiebungsfällen die BE-Berechnung durch (mit Belastung $p = 2$). Für diese Teilfunktionen können nun die Integrale über die Spannungen bestimmt werden (\textit{Gleichung 23}).

Das Loch $L_i$ kann dann später in der Gesamtfunction problemlos an die gewünschte Position „geschoben“ werden, und zwar durch entsprechende Veränderung des numerischen Vorfaktors $m$. Problemlos deswegen, weil für die Ränder der Teilfunktion die Integrale entsprechend \textit{Gleichung 23} um den gleichen, konstanten Faktor vergrößert werden. So kann für jedes Loch durch den entsprechenden Wert für den Vorfaktor die Bredtsche Gleichgewichtsbedingung für den Teilrand erfüllt werden.

Die Vorfaktoren werden durch Aufstellen eines linearen Gleichungssystems bestimmt, das bei n Rändern $n$ Gleichungen, sowie $n$ Unbekannte (die Faktoren) enthält.

\textit{Demonstration der Methode anhand des in Abbildung 47 dargestellten Querschnitts}
Der äußere Rand sei Rand 1 \((\Gamma_1)\), der Rand des ersten Lochs (mit viereckiger Geometrie) sei Rand 2 \((\Gamma_2)\), der Rand des zweiten Lochs (dreieckige Geometrie) sei Rand 3 \((\Gamma_3)\).

Dann gilt für:

- Funktion \(u_1\): \(u_1(\Gamma_1) = 0; \ u_1(\Gamma_2) = 0; \ u_1(\Gamma_3) = 0\);
- Funktion \(u_2\): \(u_2(\Gamma_1) = 0; \ u_2(\Gamma_2) = 1; \ u_2(\Gamma_3) = 0\);
- Funktion \(u_3\): \(u_3(\Gamma_1) = 0; \ u_3(\Gamma_2) = 0; \ u_3(\Gamma_3) = 1\);
- Gesamtfunction \(u\): \(u = m_1u_1 + m_2u_2 + m_3u_3\);
- für die Faktoren: \(m_1 + m_2 + m_3 = 1\);
- außerdem entsprechend für \(I_{i,\text{ges}} = m_1I_{i,u1} + m_2I_{i,u2} + m_3I_{i,u3}\) (gemeint ist das Integral über die Spannung gemäß **Gleichung 23**)

Hierbei wurden die von Null verschiedenen, konstanten Randverschiebungen beliebig zu 1 gewählt.

Berechnet man nun für jede Teilfunktion die Integrale der Randspannungen der Löcher \(I_{i,uj}\), erhält man unter Anwendung von **Gleichung 23** folgendes Gleichungssystem:

\[
\begin{align*}
(1) \quad & m_1 + m_2 + m_3 = 1 \\
(2) \quad & m_1I_{2,u1} + m_2I_{2,u2} + m_3I_{2,u3} = -2A_2 \\
(3) \quad & m_1I_{3,u1} + m_2I_{3,u2} + m_3I_{3,u3} = -2A_3
\end{align*}
\]

\(I_{2,u1}\) bedeutet: Integral über Rand 2 für Teilfunktion \(u_1\).

Dieses Gleichungssystem bestimmt eindeutig die Faktoren \(m_1\), \(m_2\) und \(m_3\), und das Problem ist gelöst.

Für den Querschnitt mit 2 Löchern erhält man mit obigen Bedingungen folgendes Gleichungssystem (für Rechnung mit 13 Elementen):

\[
Bm = brs
\]

\[
\begin{pmatrix}
1 & 1 & 1 & m_1 & 1 \\
-6.704 & 7.252 & -1.857 & m_2 & -0.077 \\
-4.354 & -2.204 & 5.896 & m_3 & -0.038
\end{pmatrix}
\]

das Gleichungssystem liefert: \(m_1 = 0.921\); \(m_2 = 0.042\); \(m_3 = 0.037\).
Wählt man die Ansatzfunktionen in obiger Form, ist die Gesamtfunktion $u$ direkt bestimmt: Mit Gleichung 25 setzt man für $u$:

\[ u(\Gamma_1) = 0; \quad u(\Gamma_2) = m_2 = 0,042; \quad u(\Gamma_3) = m_3 = 0,037; \]

Für diese Werte berechnet sich das Torsionswiderstandsmoment in gewohnter Weise: Das zweifache Volumen unter der Membran (einschließlich der Löcher) ist das gesuchte Resultat.

Die Überprüfung der Bredtschen Formel liefert bei 13 Elementen:

<table>
<thead>
<tr>
<th>Rand Nr.</th>
<th>$I_i$</th>
<th>$2A_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0,805633</td>
<td>0,769231</td>
</tr>
<tr>
<td>2</td>
<td>0,0769231</td>
<td>-0,0769231</td>
</tr>
<tr>
<td>3</td>
<td>0,0384615</td>
<td>-0,0384615</td>
</tr>
</tbody>
</table>

Rand 2 und Rand 3 erfüllen die Gleichgewichtsbedingung exakt. Dies liegt darin begründet, daß die Integrale über diese Ränder direkt in das Gleichungssystem zur Bestimmung der Faktoren eingehen (im Gegensatz zu Rand 1). Rand 1 weist den üblichen Spannungsfehler auf, der bei der Randelementmethode auftritt und mit steigender Elementzahl immer kleiner wird (siehe Abschnitt 5.1: Fehler bezüglich des Kräftegleichgewichts).

8.2.2 Hinweise zur Programmierung

Um die zur Gesamt-Verschiebungsfunktion $u$ zugehörigen Randspannungen in den Kollokationspunkten zu ermitteln, sind nun einige Erweiterungen entsprechend Abschnitt 8.2.1 an der Funktion Compute vorzunehmen. Außerdem muß ComputeTorque etwas erweitert werden, da die Randverschiebung von $u$ nun nicht mehr generell Null beträgt.

1. Erweiterungen von Compute zu ComputeForTorque

Um auch weiterhin Membranprobleme bearbeiten zu können, die nicht speziell dem Anwendungszweck der Membrananalogie bezüglich Torsion zuzuordnen sind, wurde in dieser Arbeit eine neue Funktion ComputeForTorque implementiert.
**ComputeForTorque** bestimmt die unbekannten Randdaten für Querschnitte, die beliebig viele Löcher enthalten können, unter den Bedingungen der Bredtschen Formel (Gleichung 23) für die Teilränder.

Die neue Funktion **ComputeForTorque** ist so aufgebaut:

- Berechnung der Matrizen \( G \) und \( H \), sowie des Lastvektors \( p \), die für alle Teilfunktionen \( u_i \), aus denen dann die (endgültige) Gesamtfunction ermittelt wird, gleich sind.

- Faktorisierung der Matrix \( G \), dabei wird die vorgenommene Pivotisierung zwischen- gespeichert. Dieser Rechenschritt muß nur einmal für alle Teilfunktionen ausgeführt werden. Durchgeführt mit Hilfe von **SolveScaledRowGaussPart1**.

- Schleife über die Löcher:
  - **Erzeugen** und **Berechnen** der Teilfunktionen.
    - **Erzeugen**: Entsprechend obiger Beschreibung werden die Randverschiebungen zu Null, bzw. Eins gesetzt. Diese Aufgabe übernimmt die Funktion **SetVectorU**. \( G \), \( H \) und \( p \) wurden bereits außerhalb der Schleife bestimmt.
    - **Berechnen**: mit **SolveScaledRowGaussPart2** wird der Vektor der Randspannungen (in den Kollokationspunkten) für die Teilfunktion bestimmt. Mit Hilfe dieser Spannungen können dann mit **IntegrateP** die Integrale über die Randspannungen für alle Löcher bestimmt werden. Diese Werte werden dann in die Matrix \( B \) eingetragen, deren Dimension der Anzahl der Löcher entspricht.
  - Berechnen des negativen zweifachen Flächeninhalts der Löcher, Eintrag in Vektor \( brs \).
  - Lösen des Gleichungssystems \( Bm = brs \) zur Bestimmung der Faktoren \( m_i \) mit **SolveScaledRowGauss**.
  - Berechnen der endgültigen Gesamtfunction:
    - Aufbau des neuen Vektors \( \hat{u} \) der Randverschiebungen, Bestimmen der Randspannungen in den Kollokationspunkten mit **SolveScaledRowPart2**. Hier werden wieder die bereits bestimmten \( G \), \( H \) und \( p \) verwendet.

Nun kann (wie gewohnt von den Querschnitten ohne Löcher) die Bestimmung des Torsionswiderstandsmoments über **ComputeTorque** erfolgen. **ComputeTorque** muß hierfür noch leicht erweitert werden, siehe folgende Seite.
Auszug aus `ComputeForTorque`:

```cpp
void CExColloc::ComputeForTorque()
{
    // Berechnung der Matrizen H und G, sowie Vektor p:
    ComputeHGP(GaussPoints);
    ...

    // Matrix G faktorisieren, Pivotisierung merken
    SolveScaledRowGaussPart1(*G,pivots,Size);

    // Schleife über die Teilränder
    while (k<=count)
    {
        // Vektor u setzen
        SetVectorU(k);
        // rechte Seite ausrechnen: rs=H*u-p
        for (i=1;i<=Size;i++)
        {
            rs[i-1]=-p[i-1];
            for (j=1;j<=Size;j++)
                rs[i-1]+=(*H)(i,j)*u[j-1];
        }
        // Gleichungssystem lösen (Teil 2) mit faktorisierter Matrix und Pivots
        SolveScaledRowGaussPart2(*G,rs,t,pivots,Size);
        // Spannung integrieren für alle Ränder
        IntegrateT(col);
        // in Matrix B eintragen:
        for (i=1;i<count;i++) B(i+1,k)=col[i];
        // 1. Zeile von Matrix erzeugen ( LGS 1.Zeile: m1 + m2 + ... + m3 = 1 )
        // (Vorausgesetzt: 1. Rand ist kein Loch)
        B(1,k)=1;
        k++;
    }

    // 2x umschlossene Fläche berechnen für alle Ränder und negativ nehmen:
    IntegrateP(brs);
    for (i=1;i<count;i++) brs[i]=-brs[i];
    brs[0]=1; // (hier wieder 1. Zeile)
    // LGS lösen für Bestimmung der Niveaus:
    SolveScaledRowGauss(B,brs,col,count);
    // Endgültiges Modell rechnen mit entgültigen Randverschiebungen
    ...

    // Verschiebungsvektor u aufbauen
    ...

    // rechte Seite berechen (H*u-p):
    ...

    // LGS lösen für endgültige Randpannungen t:
    SolveScaledRowGaussPart2(*G,rs,t,pivots,Size);
    // aufräumen:
    delete[] ...  
    ...
}
```
Fazit:

Sowohl die Bestimmung der Matrizen $G$ und $H$ mit $\text{ComputeHGP}$, als auch die Faktorisierung der Matrix $G$ konnten außerhalb der Schleife über die Ränder, bzw. Teilfunktionen angeordnet werden. (Die Lösung des Gleichungssystems mit $\text{SolveScaledRowGauss}$ wurde hierzu in zwei Teilschritte zerlegt: Die Faktorisierung der Matrix und das Vorwärts- und Rückwärtseinsetzen mit Berücksichtigung der rechten Seite des LGS.) $\text{ComputeHGP}$ sowie Faktorisierung machen auch den Haupt-Rechenzeitaufwand von $\text{ComputeForTorque}$ aus. Demzufolge wächst die benötigte Rechenzeit mit zunehmender Lochzahl nur unmerklich gegenüber Querschnitten ohne Löcher an. Maßgebend ist immer die Anzahl der Randelemente, ob sich diese auf mehrere oder lediglich einen Rand verteilen, spielt keine entscheidende Rolle.

2. Erweiterung von $\text{ComputeTorque}$

Hier sei an $\text{Gleichung 20}$ erinnert, die zur Bestimmung des Torsionswiderstandsmoments hergeleitet wurde:

$$
\frac{1}{2} I_t = \int_\Omega u(y) d\Omega = \int_\Gamma h(y) \frac{\partial h}{\partial n} (y) d\Gamma_y - \int_\Gamma h(y) t(y) d\Gamma_y + \left( -p \right) \int_\Omega h(y) d\Omega_y
$$

$I_t$ ist nun nicht mehr stets Null. $u(y)$ ist aber immerhin stückweise konstant, und zwar für jeden Teilrand, kann also vor das Integral gezogen werden. Die Berechnung vom verbleibenden Integral ist dann kein Problem mehr, da $h$ ja die gewählte Hilfsfunktion darstellt, deren Normalableitung bekannt ist. Die Integration kann analytisch erfolgen.

Ansonsten ändert sich hier nichts im Hinblick auf Querschnitte mit Löchern.

8.2.3 Ergebnisse für die Beispielquerschnitte

Dem Leser werden an dieser Stelle nur einige ausgewählte „Highlights“ präsentiert. Der Vergleich mit Referenzwerten für die Torsionswiderstandsmomente, berechnet mit FE-Methode, wurde nicht nochmals durchgeführt. Einerseits lag für das FE-Programm von Professor Dr. Holzer keine entsprechend $\text{Abschnitt 8.2}$ erweiterte Funktionalität vor. Andererseits ist auf-
Andererseits ist aufgrund der exakten Erfüllung der Bredtschen Bedingung für die Löcher auch keine Abweichung von den Ergebnissen unter Abschnitt 8.1.3 zu erwarten.

Abbildung 51: Adaption III, 33 Elemente

Abbildung 48: Adaption I, 13 Elemente

Abbildung 49: Adaption II, 17 Elemente

Abbildung 50: Adaption III, 21 Elemente

Abbildung 52: Adaption IV, 42 Elemente

Abbildung 53: Adaption V, 58 Elemente

Abbildung 54: Adaption VII, 80 Elemente
Auf Abbildung 48 bis Abbildung 55 ist der Ablauf der Randverfeinerung bei Adaption nach $u^2$-Kriterium für den Querschnitt mit zwei Löchern dargestellt. Man kann deutlich die Netzverfeinerung hin zu den Ecken und insbesondere hin zu den einspringenden Ecken erkennen.

Hier die adaptiv berechneten Torsionswiderstandsmomente für den Beispielquerschnitt mit zwei Löchern (Elementteilung für Integral über $u^2$-Fehler größer als 0,3-facher Maximalwert):

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Torque</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>4,01468</td>
</tr>
<tr>
<td>17</td>
<td>4,17832</td>
</tr>
<tr>
<td>21</td>
<td>3,81334</td>
</tr>
<tr>
<td>33</td>
<td>3,25899</td>
</tr>
<tr>
<td>42</td>
<td>3,10488</td>
</tr>
<tr>
<td>58</td>
<td>3,06270</td>
</tr>
<tr>
<td>80</td>
<td>3,00890</td>
</tr>
<tr>
<td>112</td>
<td>2,98757</td>
</tr>
<tr>
<td>136</td>
<td>2,98523</td>
</tr>
</tbody>
</table>

Für die anderen beiden Beispielquerschnitte ergibt sich (Elementteilung für Integral über $u^2$-Fehler größer als der mittlere Fehler):

Tabelle 17: HP 60x4

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Torque</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>918797</td>
</tr>
<tr>
<td>32</td>
<td>784923</td>
</tr>
<tr>
<td>40</td>
<td>749071</td>
</tr>
<tr>
<td>48</td>
<td>747054</td>
</tr>
<tr>
<td>56</td>
<td>742645</td>
</tr>
<tr>
<td>72</td>
<td>732112</td>
</tr>
<tr>
<td>80</td>
<td>730348</td>
</tr>
<tr>
<td>96</td>
<td>729304</td>
</tr>
</tbody>
</table>

Tabelle 18: Quadrat, gelocht

<table>
<thead>
<tr>
<th>Elemente</th>
<th>Torque</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>727728</td>
</tr>
<tr>
<td>136</td>
<td>727484</td>
</tr>
<tr>
<td>168</td>
<td>727144</td>
</tr>
<tr>
<td>232</td>
<td>727069</td>
</tr>
<tr>
<td>296</td>
<td>727016</td>
</tr>
<tr>
<td>400</td>
<td>727006</td>
</tr>
<tr>
<td>520</td>
<td>726983</td>
</tr>
</tbody>
</table>
9 Performance-Vergleich: Finite Elemente – Randelemente

9.1 Vergleich bezüglich Rechenzeit


Beim Dreieck ist die Randelementmethode den finiten Elementen deutlich überlegen. Bei diesem simplen Querschnitt mit gleichen Elementlängen verläuft die Konvergenzkurve schon für wenige Elemente sehr steil. Mit der Methode der finiten Elemente läßt sich aufgrund des
flacheren Verlaufes in annehmbarer Zeit bei weitem nicht die Genauigkeit erreichen, die die Boundary-Element-Methode bereits nach circa einer Minute liefert. Das adaptive Vorgehen liefert hier keine Vorteile, ist aber auch nicht dramatisch schlechter. Für höhere Genauigkeiten ist ein etwas günstigeres Verhalten zu erwarten.
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Abbildung 57: Rechenzeit-Fehler-Schaubild für T-Profil

Abbildung 58: Rechenzeit-Fehler-Schaubild für I-Profil


9.2 Vergleich weiterer typischer Eigenschaften

- **Stabilität/Sicherheit**

- **Speicherplatzbedarf**
  Dadurch, daß bei der Boundary-Element-Methode nur der Rand diskretisiert werden muß, ist der Speicherplatzbedarf für die Elementmatrizen deutlich geringer. Die FE-Netze erreichen schon für angestrebte Genauigkeiten um 1% bis 0,1% erstaunliche Dimensionen,
Dimensionen, was die Implementierung effizienter iterativer Gleichungslöser dringend erforderlich macht. Ohne besondere Maßnahmen der Matrizenverwaltung, d.h. speicherplatzsparende Speicherung, können keine sehr großen Systeme effizient verarbeitet werden: Bei einem verfügbaren Arbeitsspeicher von 16 MB (geschätzter verfügbarer Speicher auf einem durchschnittlichen 32-MB-Windows-Betriebssystem) wäre beispielsweise die maximal komplett im Arbeitsspeicher haltbare Matrix auf die Dimension $\sqrt{16 \times 1024 \times 1024 / 8} = 1448$ beschränkt (doppelte Fließkommmazahlen-Genauigkeit). Wird dieser Wert überschritten, hat dies Auslagerungen der Daten auf Festplatte zur Folge und führt zu dramatischer Verlängerung der Bearbeitungszeit.
10 Verbesserung der Ergebnisse durch Extrapolation

Nutzt man die Eigenschaft des Verfahrens der Randelementmethode, daß sich für den relativen Fehler der Berechnung mit wachsender Elementzahl ein näherungsweise geradenförmiger Verlauf im doppelt-logarithmischen Maßstab einstellt, kann mit drei berechneten Schritten $I_{\text{ber}}$ eine Extrapolation auf den erwarteten Konvergenzwert $I_{\text{ex}}$ erfolgen.

Der extrapolierte Wert $I_{\text{ex}}$ ist gleichzeitig Bezugsgröße, da er zur Berechnung des relativen Fehlers $f$ dient. Je nach Wahl der Bezugsgröße stellt sich der Verlauf des Fehlers unterschiedlich dar. Dies ist klar, da man je nach Bezugsgröße andere Werte für den Fehler $f$ erhält:

$$f = \left| \frac{I_{\text{ber}}}{I_{\text{ex}}} - 1 \right|$$

Ziel des Verfahrens ist nun, die Bezugsgröße so einzustellen, daß sich der Verlauf des Fehlers im doppelt-logarithmischen Maßstab exakt als Gerade darstellt, d.h. alle drei Berechnungspunkte sollen auf einer Geraden liegen. Siehe hierzu Abbildung 59.

Man erhält folgende vier Gleichungen, (4) unter Aufstellen der Geradengleichungen für die Punkte $(n_1f_1)$ und $(n_2f_2)$, sowie für $(n_1f_1)$ und $(n_3f_3)$:
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

(1) \( f_1 = \frac{I_{ber,1}}{x} - 1 \)

(2) \( f_2 = \frac{I_{ber,2}}{x} - 1 \)

(3) \( f_3 = \frac{I_{ber,3}}{x} - 1 \)

(4) \( \frac{\ln f_1 - \ln f_2}{\ln m - \ln n_2} - \frac{\ln f_1 - \ln f_3}{\ln m - \ln n_3} = 0 \)

Mit Hilfe dieser Gleichungen kann nun \( x (= I_{ex}) \) bestimmt werden. Nach Einsetzen von (1) bis (3) in (4) kann die resultierende Gleichung numerisch gelöst werden.

Für diesen Zweck wurde das Sekantenverfahren verwendet. Beachtet werden muß hier eine möglichst günstige Wahl des Startwertes für die Iteration.

Die entsprechende Funktion in C++ heißt \textit{FindLimit}. Sie wird in \textit{limit.h} deklariert und in \textit{limit.cpp} definiert. Übergeben werden müssen drei berechnete Widerstandsmomente \( I_{ber} \), sowie die zugehörige Elementanzahl \( n \), bzw. Anzahl der Unbekannten.

Auch bei der Methode der finiten Elemente kann die Extrapolation angewendet werden. Entsprechend dem hier beschriebenen Verfahren wurden auch die in den vorangegangenen Abschnitten verwendeten Referenzwerte für die Torsionswiderstandsmomente bestimmt.

Im folgenden sind für einige Querschnitte die extrapolierten Torsionsträgheitsmomente aus jeweils drei mit FEM bzw. BEM berechneten Werten tabellarisch zum Vergleich angegeben. Die Netzverfeinerung erfolgte bei beiden Methoden gleichmäßig linear.

Tabelle 19: Dreieck

<table>
<thead>
<tr>
<th>Elemente</th>
<th>( I_{ber} )</th>
<th>( I_{ex} )</th>
<th>Unbekannte</th>
<th>( I_{ber} )</th>
<th>( I_{ex} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>282572</td>
<td>1770</td>
<td>280213</td>
<td>280590,25</td>
<td>280593,66</td>
</tr>
<tr>
<td>96</td>
<td>280928</td>
<td>4846</td>
<td>280453</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>280649</td>
<td>11189</td>
<td>280531</td>
<td>280590,25</td>
<td>280593,66</td>
</tr>
<tr>
<td>384</td>
<td>280602</td>
<td>22544</td>
<td>280561,8</td>
<td>280593,66</td>
<td></td>
</tr>
<tr>
<td>768</td>
<td>280594</td>
<td>280592,36</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Berechnung von St.-Venantschen Torsionsträgheitsmomenten mit der Randintegralmethode

Tabelle 20: T-Profil

<table>
<thead>
<tr>
<th>Elemente</th>
<th>( I_{ber} )</th>
<th>( I_{ex} )</th>
<th>Unbekannte</th>
<th>( I_{ber} )</th>
<th>( I_{ex} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>100548</td>
<td>1510</td>
<td>93447,94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>95736,6</td>
<td>4336</td>
<td>93850,19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>94576,1</td>
<td>10012</td>
<td>93999,46</td>
<td>94158,20</td>
<td></td>
</tr>
<tr>
<td>352</td>
<td>94285,4</td>
<td>17695</td>
<td>94057,80</td>
<td>94162,88</td>
<td></td>
</tr>
<tr>
<td>704</td>
<td>94204,9</td>
<td>1120</td>
<td>94074,07</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 21: I-Profil

<table>
<thead>
<tr>
<th>Elemente</th>
<th>( I_{ber} )</th>
<th>( I_{ex} )</th>
<th>Unbekannte</th>
<th>( I_{ber} )</th>
<th>( I_{ex} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>172,596</td>
<td>1139</td>
<td>56,42350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>95,1128</td>
<td>4056</td>
<td>58,472567</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>67,6830</td>
<td>651175</td>
<td>59,3562</td>
<td></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>61,1338</td>
<td>59,079643</td>
<td>59,3267</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td>59,7174</td>
<td>59,326543</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 22: Gelochtes Quadrat (Randverschiebung =0)

<table>
<thead>
<tr>
<th>Elemente</th>
<th>( 2^*V_{ber} )</th>
<th>( 2^*V_{ex} )</th>
<th>Unbekannte</th>
<th>( 2^*V_{ber} )</th>
<th>( 2^*V_{ex} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>16,1586</td>
<td>1272</td>
<td>15,412888</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>15,6986</td>
<td>3739</td>
<td>15,492128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>15,5919 15,559511</td>
<td>8464</td>
<td>15,521922 15,558583</td>
<td></td>
<td></td>
</tr>
<tr>
<td>384</td>
<td>15,5660 15,557698</td>
<td>15595</td>
<td>15,533936 15,551497</td>
<td></td>
<td></td>
</tr>
<tr>
<td>768</td>
<td>15,5590 15,556407</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Die bisherigen Untersuchungen an den zahlreichen Querschnitten haben gezeigt, daß sich selbst für adaptive Verfeinerung der Randeinteilung eine näherungsweise geradenförmige Konvergenz im doppelt-logarithmischen Maßstab einstellt. Es liegt also nahe, auch für die adaptive Netzverfeinerung eine Extrapolation vorzunehmen.

Hierzu wurde die Funktion \textit{AdaptionLimit} implementiert, die sowohl für gelochte als auch für ungelochte Querschnitte verfügbar ist. \textit{AdaptionLimit} verfeinert das Netz adaptiv so lange, bis die angegebene Elementzahl erreicht wurde. Mindestens werden jedoch 6 Adaptionsschritte ausgeführt. Dann erfolgt die Extrapolation mit drei berechneten Torsionsträgheitsmoment-Zwischenwerten. Kann keine Konvergenz eingestellt werden, was in sehr ungünstigen Fällen eintreten kann (dann, wenn der Verlauf des Fehlers infolge der adaptiven Verfeinerung zu stark von der idealisierten Geraden im doppelt-logarithmischen Maßstab abweicht), wird der letzte Wert der Adaption zurückgegeben.

*Tabelle 23: AdaptionLimit für 150 Elemente*

<table>
<thead>
<tr>
<th>Querschnitt</th>
<th>Rechenzeit (P166) [s]</th>
<th>Torsionswiderstandsmoment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dreieck</td>
<td>7,0</td>
<td>280556</td>
</tr>
<tr>
<td>T-Profil</td>
<td>9,7</td>
<td>94205</td>
</tr>
<tr>
<td>I-Profil</td>
<td>10,7</td>
<td>58,899</td>
</tr>
</tbody>
</table>
11 Einbettung der BE-Methode in eine graphisch-interaktive Umgebung

Die Klassen der Randelementmethoden, sei es für Kollokation, Galerkin oder auch die erweiterte Kollokation für gelochte Querschnitte haben gemeinsam, daß die verwendeten Strukturen und Funktionen nur grundlegende Eigenschaften zur Manipulation und Darstellung aufweisen. Der Schwerpunkt liegt hier ganz klar auf möglichst effizienter Berechnung bereits im Speicher befindlicher Modelle.


Für den Anwender, der sich nicht tiefer in die hier vorgestellten Themen einarbeiten möchte, sondern lediglich an einer schnellen Lösung konkreter Torsionswiderstandsprobleme interessiert ist, wurde im Rahmen dieser Diplomarbeit das Programm „BEM Torque Calculation“ entwickelt. BEM Torque Calculation bettet die Randelementmethode in eine graphisch-interaktive Benutzeroberfläche ein.

Das Programm ermöglicht die Eingabe von Querschnitten mit beliebig vielen Löchern per Maus und die Berechnung der Torsionswiderstands­momente. Als kleiner „Bonus“ können auch die übrigen Querschnittswerte berechnet werden: Flächeninhalt, Schwerpunkt, sowie die Flächenträgheitsmomente zweiten Grades.

Es folgt nun eine Beschreibung der zentralen Gedanken bei der Entwicklung.

11.1 Die Basisbibliothek CIALib

Als Basis für die graphisch-interaktive Umgebung wurde die sog. „CIALib“ verwendet. CIALib (classes, interactive) ist eine Bibliothek zur Erstellung einfacher graphisch-interaktiver Anwendungen unter Windows. Sie wurde unter der Leitung von Herrn Prof. Holzer am Fachgebiet für Informatik im konstruktiven Ingenieurbau entwickelt, und befindet sich in ständiger Erweiterung.

Möchte der Leser mehr über die CIALib erfahren, ist eine Dokumentation in Form von Skriptauszügen am Fachgebiet evtl. auf Anfrage erhältlich.

**Grundkonzepte der CIALib**

- die Klassenbibliothek folgt dem objektorientierten Paradigma, d.h. die Implementierung der Programmenteile folgt in Form von Klassen nach einer systematischen Hierarchie. So wird hohe Übersichtlichkeit gewährleistet.


- Die gesamte Oberfläche basiert auf Fenstern. So sind sowohl Knöpfe und Menüleisten, als auch Eingabefelder und Ausgabebereiche stets Fenster. Demzufolge leiten sich auch die entsprechenden Klassen von derselben Basisklasse für Fenster CIAWnd ab.
11.2 Aufbau des graphisch-interaktiven Programms

Die Applikation soll zwei Hauptaufgaben wahrnehmen: Die graphische Ein- und Ausgabe von Querschnitten, sowie die Steuerung der Berechnungen.

Da die CIALib nach dem Model-View-Controller-Prinzip aufgebaut ist, ist hier bereits eine geordnete Klassen-Grundstruktur vorhanden. Um die anwendungsspezifischen Erweiterungen ins Programm einbringen zu können, werden neue Klassen von den bestehenden abgeleitet.


Das Modell der Randelementmethode wird durch die Klasse BEMMod beschrieben. Sie wurde von CIAModel abgeleitet. CIAModel weist zahlreiche virtuelle Funktionen auf, die durch Überladen an die gewünschten Eigenschaften angepaßt werden können. Als wichtigste Datenmember enthält BEMMod die Klassen CVBModel (für die Erfassung der Querschnittsgeometrie) und CExColloc (für die Berechnungen).

CVBModel, CVBNode und CVBEdge sind die Klassen der Randelementmethode. Sie wurden von CBModel, CBNode und CBEdge abgeleitet, und haben zusätzlich durch Ableitung von der Objektklasse CVisible nun Funktionen, die zur graphischen Darstellung und interaktiven Bearbeitung dienen; natürlich innerhalb der Vorgaben durch die CIALib. Die Knoten und Kanten des Randmodells sind nun nicht mehr reine Daten, sondern graphische Objekte, die per Mausklick selektiert, verschoben oder manipuliert werden können.

Abbildung 60: Klassenhierarchie BEM Torque Calculation
11.3 Programmbeschreibung: *BEM Torque Calculation*

11.3.1 Erstellen neuer Modelle


Abbildung 61: *BEM Torque Calculation* / Neues Polygon einfügen

Bestehende Polygonzüge können mit **Add node** und **Delete edge** manipuliert werden. Hierzu wird die Kante, die verändert werden soll, mit der Maus durch Drücken der linken Maustaste selektiert. Die Kante erscheint darauf blau markiert. Drücken von **Delete edge** entfernt dann die gewünschte Kante. **Add node** ermöglicht das Hinzufügen weiterer Ecken, die die jeweils markierte Kante dann teilen. Siehe hierzu *Abbildung 62*.

Abbildung 62: BEM TC / Erst Kante selektieren, dann Knoten einfügen

Abbildung 63: BEM TC / Dialogbox für Kante


Abbildung 64: BEM TC / Löcher hinzufügen
11.3.2 Anpassen der Anzeige


Das Raster kann mit **Toggle grid** ein- und ausgeschaltet werden. Anklicken des **Set grid**-Buttons öffnet eine Dialogbox, die die Eingabe einer neuen Rasterweite erlaubt.

Abbildung 65: *BEM TC* / Zoom-Funktionen +/- und all

11.3.3 Laden und Speichern von Modellen

Mit **Save** und **Load** kann das Modell abgespeichert, bzw. wieder eingelesen werden. Der Dateiname muß im Dialogfenster explizit angegeben werden. Die Speicherung erfolgt als Textfile im ASCII-Format, die Datei läßt sich also auch extern mit einem Editor bearbeiten. Gespeichert und geladen wird aus dem Verzeichnis, aus dem das Programm gestartet wurde.

11.3.4 Ausführen von Berechnungen

Mit dem Anklicken des Buttons **Compute** werden die Standard-Flächenkenngrößen des Querschnitts berechnet:

- **A**: Flächeninhalt.
- **ys/zs**: Koordinaten des Schwerpunktes bezüglich dem globalen Eingabe-Koordinatensystem.
- **Iyy/Izz/Iyz**: Flächenträgheitsmomente 2. Grades bezüglich den Schwerachsen.
Die Ausgabe der Ergebnisse erfolgt im Ergebnisfenster Result Window. Siehe Abbildung 66.

Zur Berechnung des Torsionsträgheitsmoments stehen drei Operationen zur Verfügung. In jedem Fall erfolgt die Berechnung jedoch adaptiv.

Mit Adaption step wird ein einzelner neuer Adaptionsschritt ausgeführt. Im Dialogfenster können zunächst einige Einstellungen vorgenommen werden, um die Elementteilung zu beeinflussen: Ein Faktor, der zwischen 0 und 1 liegen muß, sowie ein Schalter, der entscheidet, ob als Vergleichswert für die Teilung der Maximal- oder Durchschnittswert genommen werden soll, können verändert werden.

Günstige Werte sind hier:

Divide for: 1 Crit: 1 [AVERAGE]
oder Divide for: 0.3 Crit: 0 [MAX]

Nach Anklicken von OK wird der Adaptionsschritt ausgeführt.

Im Ergebnisfenster wird neben dem errechneten Torsionswiderstand der Fehler bezüglich des Kräftegleichgewichts angezeigt. In eckigen Klammern ist die Anzahl der Elemente des per Adaptionsschritt verfeinerten Randmodells angegeben.

Adaption erlaubt die Ausführung mehrerer Adaptionsschritte. Dabei kann die Netzverfeinerung schrittweise beobachtet werden. Als Abbruchkriterium kann in der Dialogbox zusätz-
Dialogbox zusätzlich zu oben genannten Einstellungen eine Elementanzahl angegeben werden. Wird diese Elementanzahl erreicht, ist die Adaption beendet.

Adaption limit ist die am einfachsten zu handhabende Operation zur Bestimmung des Torsionswiderstands. Es muß lediglich die Anzahl der Elemente angegeben werden, bis zu welcher mindestens adaptiv verfeinert werden soll. Im Result Window wird dann das durch Extrapolation verbesserte Endergebnis angezeigt.

11.3.5 Sonstige Funktionen

Mit Unite kann der Rand des aktuellen Modells auf ein möglichst geringe Elementzahl reduziert werden, ohne die Geometrie zu verändern. Berechnete Querschnitte lassen sich auf die Ausgangsform ohne Randverfeinerung zurückführen.

Mit Divide können lineare Elementteilungen vorgenommen werden. Diese Funktion wird derzeit im Prinzip nicht benötigt, da die Berechnung im Programm stets adaptiv erfolgt, und die Elementteilung hierbei automatisch vorgenommen wird.

Clear löscht das gesamte Modell.

12 Zusammenfassung

Die Randintegralmethode hat in den letzten Jahren gegenüber der populäreren Methode der finiten Elemente weniger Berücksichtigung erfahren. Diese Diplomarbeit zeigt jedoch, daß für bestimmte Anwendungsgebiete (d.h. im konkreten Falle die Membrananalogie bei Torsionsproblemen) die Methode der Randelemente den finiten Elementen im direkten Vergleich überlegen ist. So können St.-Venantsche Torsionswiderstandsmomente polygonal berandeter Querschnitte mit großer Effektivität numerisch bestimmt werden.


13 Anhang

A Inhalt der Begleit-CD
B Literaturhinweise

[1] Hartmann, F.: Methode der Randelemente


Kopplung mit der Finite-Elemente-Methode zur elastoplastischen Strukturanalyse

[4] Bronstein; Semendjajew: Taschenbuch der Mathematik


C Abbildungsverzeichnis

Abbildung 1: Randwertproblem Gerade ................................................................. 2
Abbildung 2: Beispiel für eine beliebige Membran ............................................. 3
Abbildung 3: Notation .......................................................................................... 4
Abbildung 4: Reduziertes Membranproblem ....................................................... 6
Abbildung 5: Symbolische Darstellung der Ringstruktur .................................... 9
Abbildung 6: Teilfunktionen; Beispielverlauf ..................................................... 10
Abbildung 7: Transformation .............................................................................. 13
Abbildung 42: Geometrie gelochtes Quadrat................................................................. 56
Abbildung 43: Geometrie rechteckiges Hohlprofil....................................................... 56
Abbildung 44: Geometrie 6-Eck mit 2 Löchern............................................................ 56
Abbildung 45: Datenmodell-Beispiel: gelochte Querschnitte ...................................... 57
Abbildung 46: Fehler gegen Referenzwert für Lochquerschnitte mit Randverschiebung =0.. 62
Abbildung 47: Beispiel für Horizontalverschiebung der Löcher................................. 64
Abbildung 48: Adaption I, 13 Elemente ................................................................. 71
Abbildung 49: Adaption II, 17 Elemente ................................................................. 71
Abbildung 50: Adaption III, 21 Elemente................................................................. 71
Abbildung 51: Adaption III, 33 Elemente ................................................................. 71
Abbildung 52: Adaption IV, 42 Elemente ................................................................. 71
Abbildung 53: Adaption V, 58 Elemente ................................................................. 71
Abbildung 54: Adaption VII, 80 Elemente ............................................................... 71
Abbildung 55: Adaption VIII, 112 Elemente ........................................................... 72
Abbildung 56: Rechenzeit-Fehler-Schaubild für Dreieck........................................... 73
Abbildung 57: Rechenzeit-Fehler-Schaubild für T-Profil.......................................... 75
Abbildung 58: Rechenzeit-Fehler-Schaubild für I-Profil.......................................... 75
Abbildung 59: Einstellen der Bezugsgröße............................................................... 78
Abbildung 60: Klassenhierarchie BEM Torque Calculation....................................... 84
Abbildung 61: BEM Torque Calculation / Neues Polygon einfügen........................... 85
Abbildung 62: BEM TC / Erst Kante selektieren, dann Knoten einfügen................... 86
Abbildung 63: BEM TC / Dialogbox für Kante ........................................................... 86
Abbildung 64: BEM TC / Löcher hinzufügen ......................................................... 86
Abbildung 65: BEM TC / Zoom-Funktionen +/- und all......................................... 87
Abbildung 66: BEM TC / Compute........................................................................... 88
Abbildung 67: BEM TC / Adaption........................................................................... 89